操作系统研讨课

蒋德钧

Fall Term 2019-2020

email: jiangdejun@ict.ac.cn
office phone: 62601007

中国科学院大学
University of Chinese Academy of Sciences

Lecture 6 File System

2019.12.16

中国科学院大学
University of Chinese Academy of Sciences

Schedule

- Project 5 due
- Project 6 assignment
  - final project, no final exam for this course

Project 5 Due

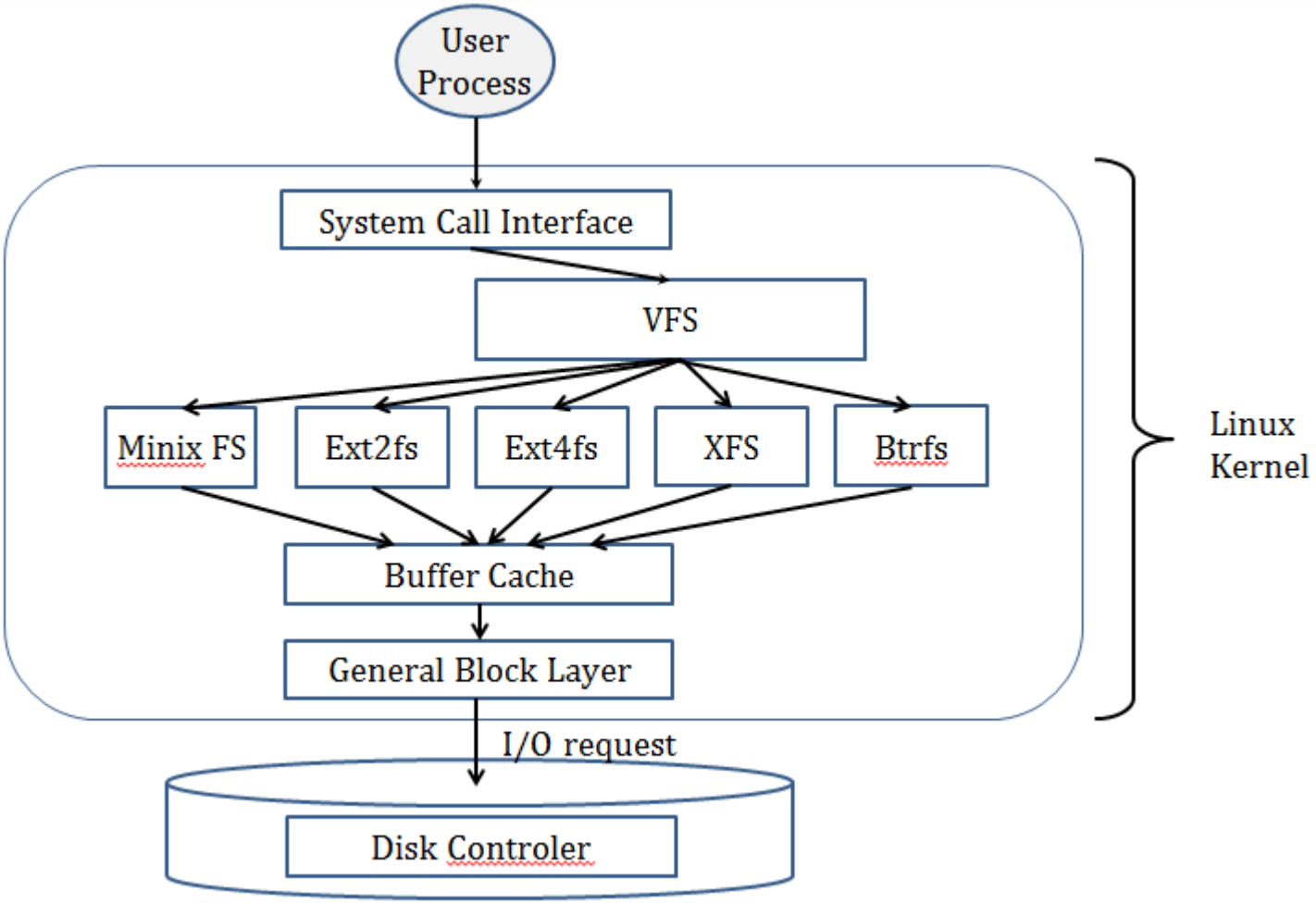
- Examining P5
  - If you finish task3, we only test task3.
  - Otherwise, please show your finished task(s) to us

Project 6 File System

- Requirement
  - Implement a simple file system
    - Disk and File system metadata management
    - Hierarchical directory structure
    - Common file system operations
      - mkfs/mkdir/rmdir/ls/statfs/cd
      - touch/cat/open/read/write/close

Project 6 File System

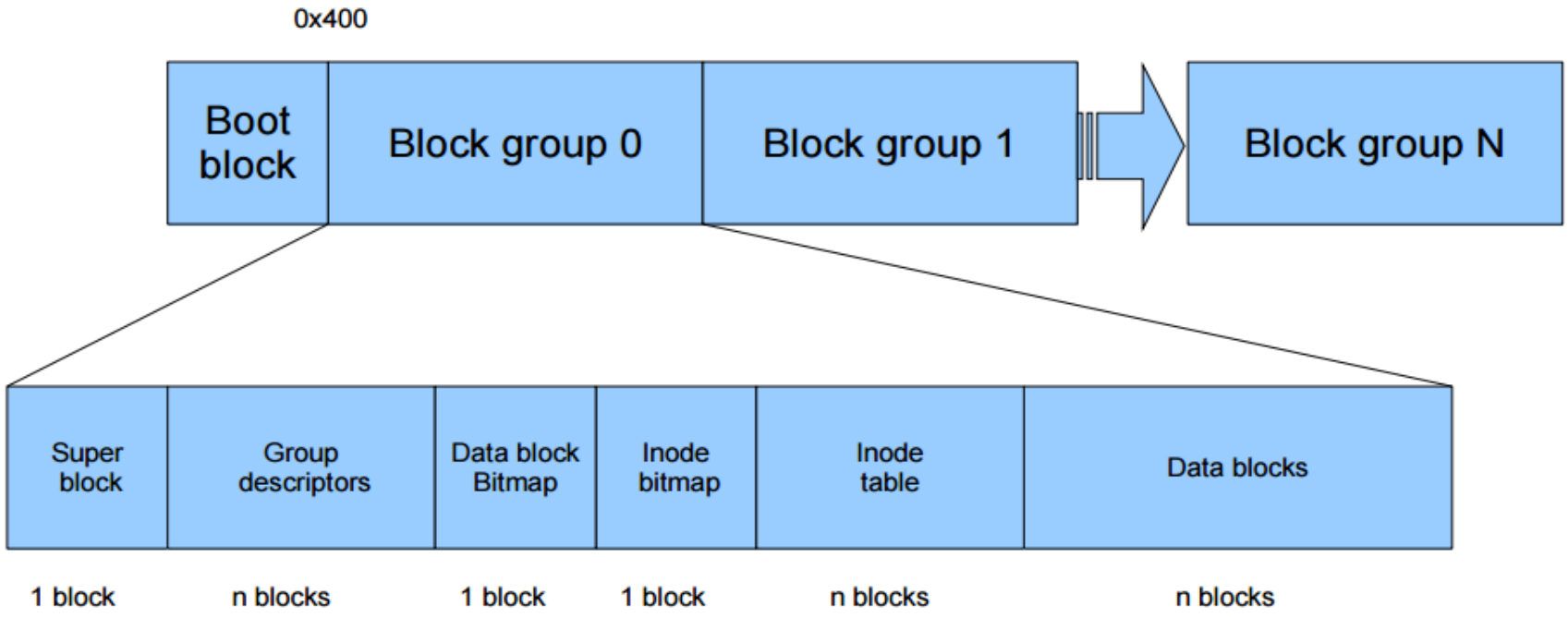
- File system



Project 6 File System

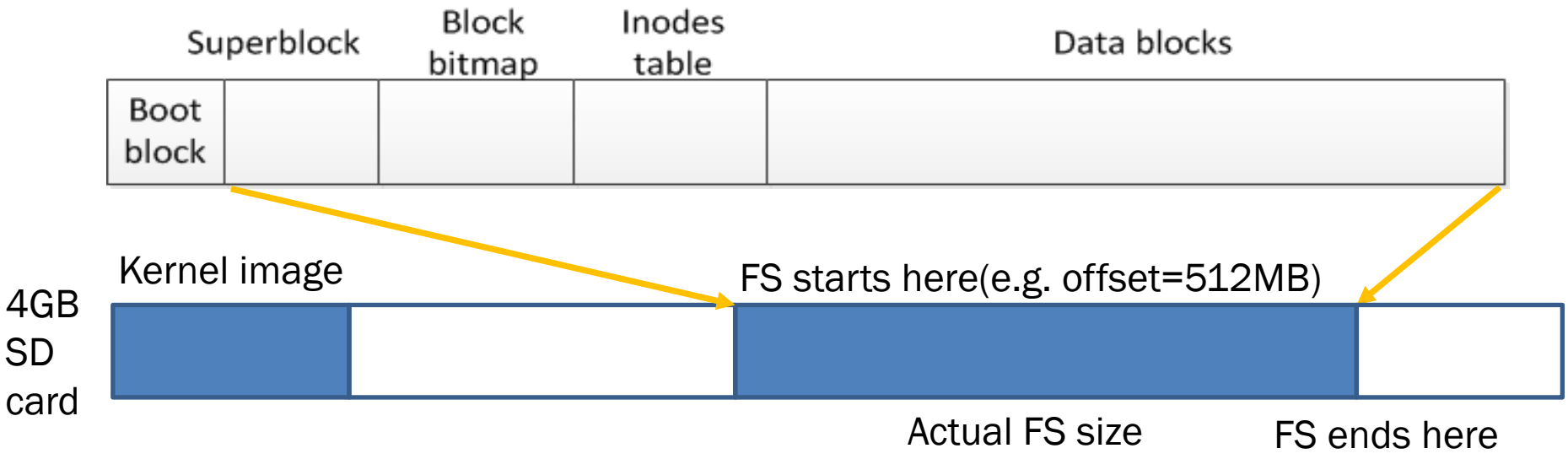
- Disk layout
  - How to manage the disk?

An example: ext2 FS disk layout



Project 6 File System

- Disk layout
  - Design your own disk layout
  - Note that
    - It is necessary to leave the space for your kernel image
    - Choose the starting block in SD card for your FS



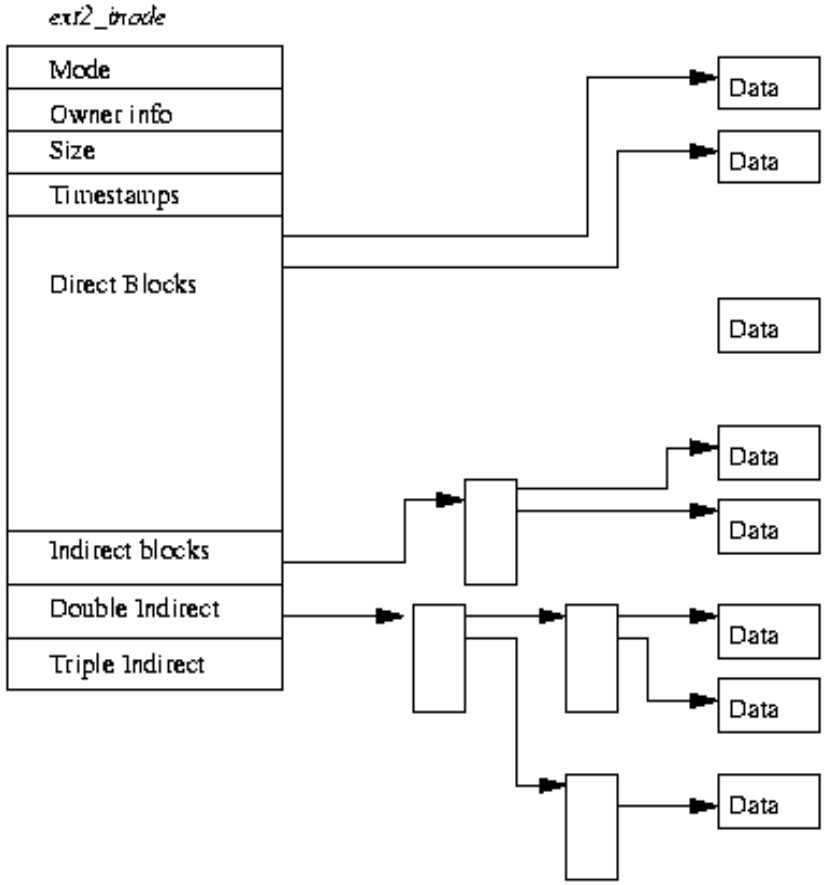
Project 6 File System

- Superblock

– Metadata to describe the structure of the file system, e.g.
  - Size
  - Num. of inodes
  - Num. of data blocks
  - Start address of inodes
  - Start address of data blocks
  - Magic number: used to judge an existing FS or not

Project 6 File System

- Inodes
  - Metadata to describe file/directory
    - Mode
    - Size
    - Timestamps
    - Addresses of data blocks
      - Direct index
      - Indirect index: you need indirect index for bonus



Project 6 File System

- File descriptor table
  - Keeping information of opening files, e.g.
    - file descriptor number (fd)
    - inode number
    - fd availability
    - Current seek position

Project 6 File System

- Directories (dentry)
  - A special file containing list of files and directories
  - Including file name and inode number
  - Always has two entries
    - Current directory “.”
    - Parent directory “..”
  - Consider carefully about indexing the contents within dentry (you need this when doing bonus for *find*)

Project 6 File System

- FS operations – mkfs
  - Write FS metadata, e.g. superblock, block info, inode table etc.
  - Create root directory and file descriptor table
  - Initialize these data structures in memory
  - Note that when mkfs is called, the file system may already be created
    - You need to judge if the file system already exists. Do not destroy the existing file system

Project 6 File System

- FS operations – mkfs
  - Note that
    - When the OS kernel starts, the kernel needs to read FS superblock, and initialize its in-memory structure
    - In case of no existing FS, the kernel invokes mkfs to create file system
    - Actually, we combine *mkfs* and *mount* operations in one step here

Project 6 File System

- FS operations – statfs
  - Return metadata info of a file system
  - When calling statfs, the basic info of your file systems are displayed

Project 6 File System

- File operations – open
  - Open an existing/ a non-existing file
  - Flags indicating the operation mode
    - read_only, write_only, rd_wr
  - Return a file descriptor by a successful call
- File operations – close
  - Free file descriptor
  - How to deal with the space occupied by the file?
    - Free the space or Not?

Project 6 File System

- File operations
  - mknod(touch): create a file
  - read: read bytes from an open file
  - write: write bytes into an open file

Project 6 File System

- Directories operation – mkdir
  - Create a directory
    - Create an entry in parent directory
    - Create two directories “.” and “..”
    - At least, your file system needs to support two-level directories
- Directories operation – rmdir
  - Remove a directory
- Directories operation – readdir
  - List all contents within an directory

Project 6 File System

- Tips on implementing file system
  - Use the SD card as the disk for your file system
  - Pay attention to inode/superblock alignment when writing to disk
  - Pay attention to dealing with manipulating existing files/directories
  - You need to implement shell commands for FS as well as their corresponding system calls
  - Please reads the guiding book carefully

Project 6 File System

- Step by step
  - Step1
    - Design and implement mkfs and statfs. Pay attention to various FS metadata, e.g. superblock, inode, file descriptor, block allocation
    - Design and implement directory operations, including mkdir, rmdir, ls, and cd
  - Step 2
    - Design and implement file operations, including touch, cat, open, read, write, and close

Project 6 File System

- Requirements for design review (40 points)
  - What is the disk layout in your design?
  - Show the structures of your FS metadata, including superblock, inode, dentry, and file descriptor
  - How large is your file system in terms of the disk managed by your FS? How many files and directories do you file system support?
  - What do you do when initializing a file system?
  - Given an operation, for example *ls /home/student*, How do you handle path lookup?

Project 6 File System

- Requirements of developing (60 points)
  - Implement mkfs, statfs, mkdir, rmdir, ls, cd (30)
  - Implement open/mknod/read/write/close (30)

Project 6 File System

- Bonus 1 (1 points)
  - Implement a few more FS operations, including
    - find
    - rename
    - ln (hard link)
    - ln –s (symbol link)

Project 6 File System

- Bonus 2 (1 points)
  - Support large file
  - Use indirect indexing to support a file at least with the size of 256MB

Project 6 File System

- P6 schedule
  - P6 design review: 23rd Dec.
  - P6 due: 30th Dec.
  - Second due: 3rd Jan. 2020

Big bonus

- Make all your modules work together
  - Use P5 to transfer an executable file from your laptop to the card, whose source code is provided with this Project.
  - Invoke file system APIs (P6) to write this executable file into SD card. Make sure this file is actually persisted

Big bonus

- Make all your modules work together
  - Use shell(P3) to execute ./your-file-name to run the test program
  - When running the test program, you need to parse ELF format(P1), and setup user-space stack, data segment, and text segment(P4), finally spawn(P2) this process

Big bonus

- You are strongly encouraged to implement this bonus if you are interested in system as well as you are available
- Please feel free to ask any questions about this bonus
- Once you finish this bonus, you will get 100 for this whole course

Big bonus

- Big bonus schedule
  - Design review: 23rd Dec. (Optional)
  - First Due: 3rd Jan. 2020
  - Final Due: 15th Jan. 2020