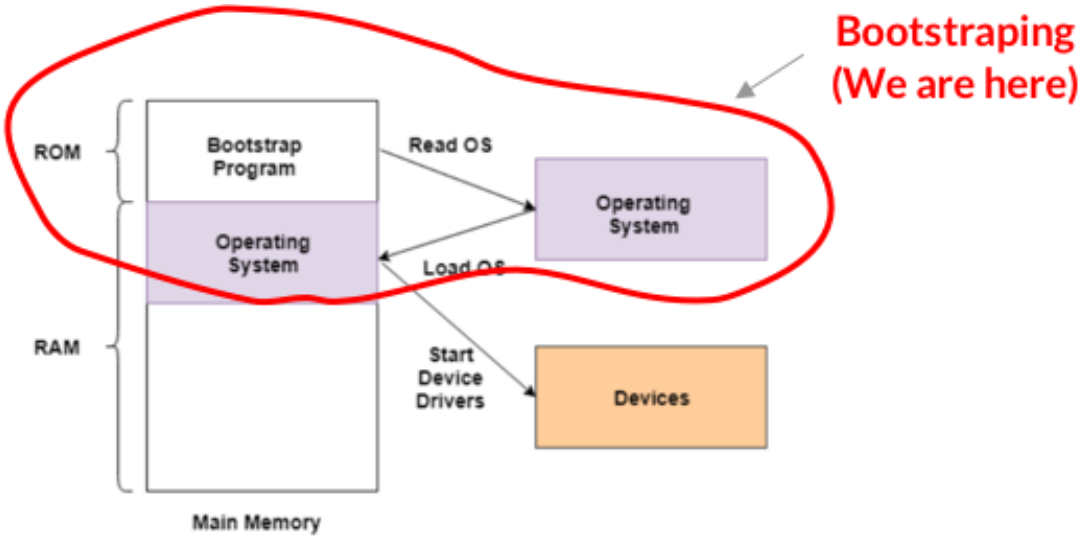
CSC369 Week 2 Notes

Hyungmo Gu

May 20, 2020

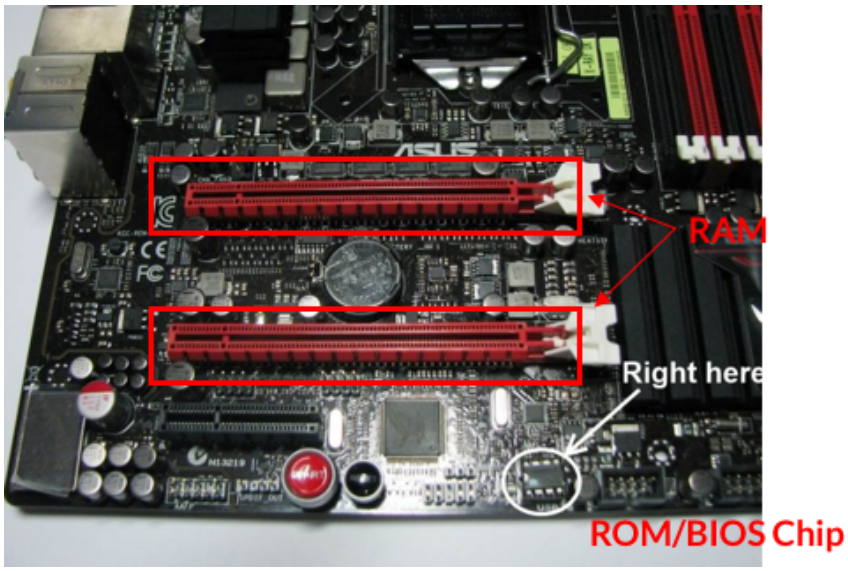
1 System Calls

- Bootstraping
  - Bootstraping

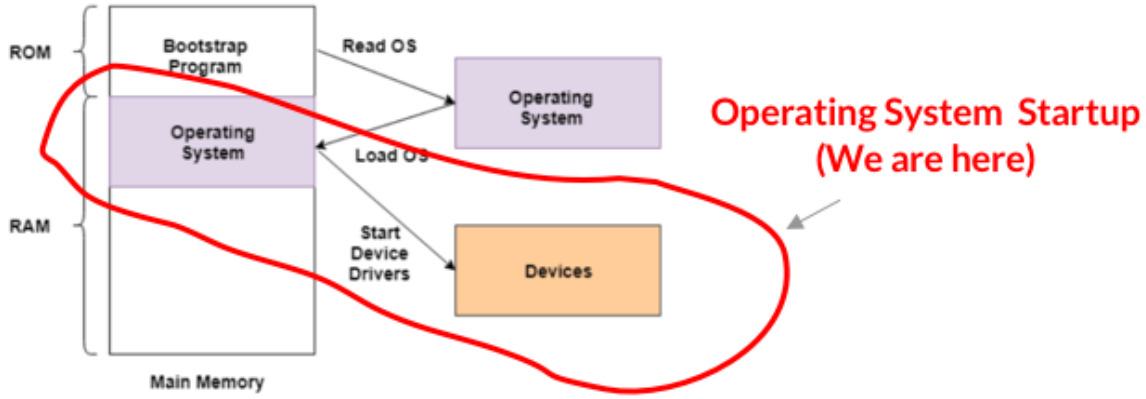


    - executes **Bootstrap Program**
      - is the first code that runs when the computer system is started
    - Entire operating system depnds on the bootstrap program to work correctly
    - Locates and loads kernel (code of operating system) onto RAM
      - kernel = code of the operating system
      - kernel is in HDD
    - Bootstrap program is in ROM
  - ROM
    - is called **read-only-memory**
    - Is also called **BIOS chip** (Basic Input/Output System)

    - is non-volatile
    - is stored in motherboard



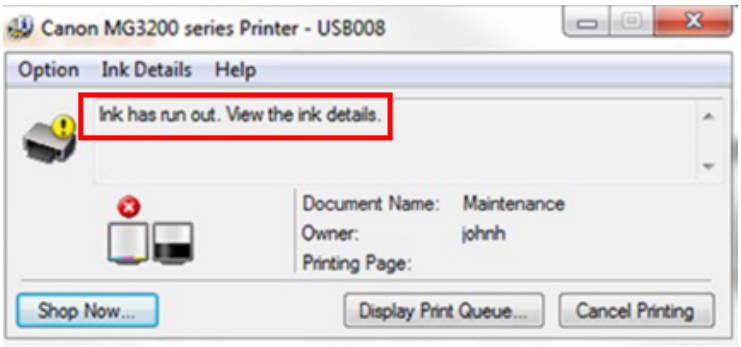
- Operating System Startup



    - Initializes OS
        - Initialize internal data structures
        - Create first process
        - Switch mode to user and start runing first process
        - Wait for something to happen

- Requesting OS Services
    - Some services offered by OS are:
        - Program execution
            - Loading program to memory and executing program

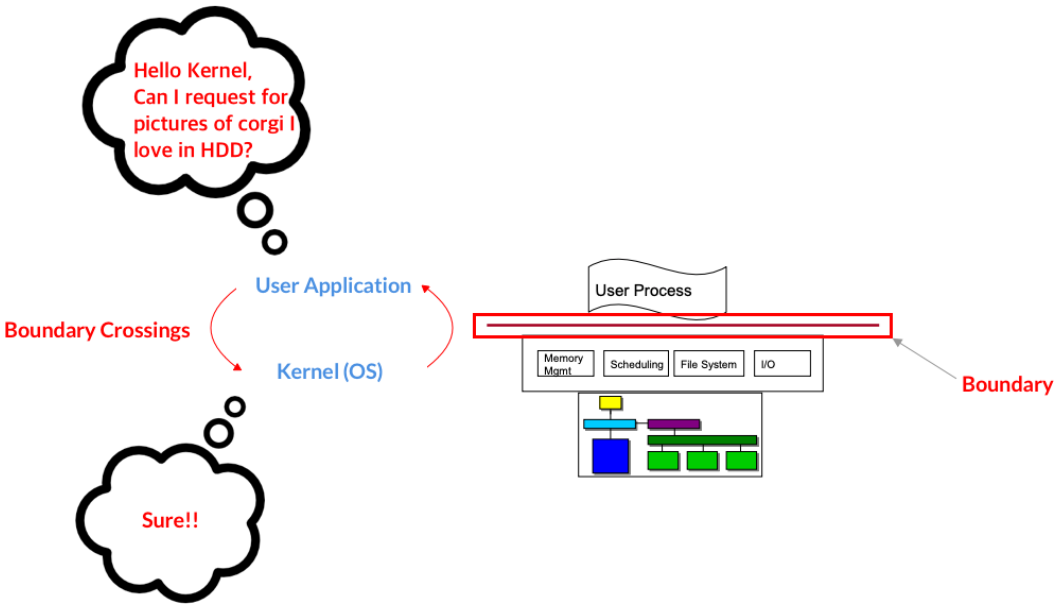
    * I/O operations
        · Keyboard, mouse, speaker
    * File system manipulation
        · Reading and writing files and directories
    * Error Detection
        · Error that pops when printer ink is empty



  – Operating system and user programs are isolated
  – How do they communicate?

- Boundary Crossings

  – Boundary
    * Is the line between user applications and kernel
    * Data is difficult to move back and forth between this line
  – Boundary Corssings
    * Is the communication that occurs between a program and kernel



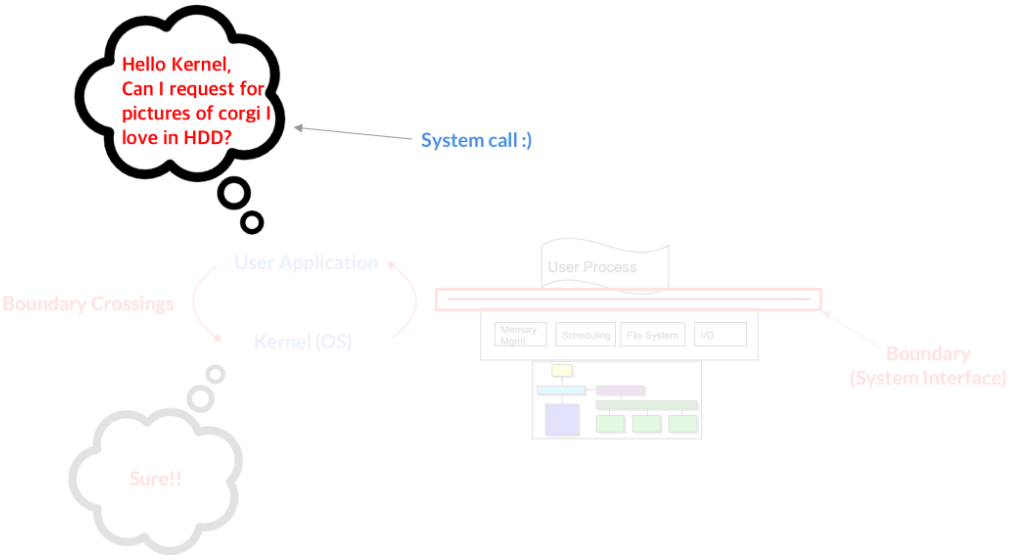
    - Communication occurs by sending data from one program into kernel, and then back
  - More can be found here

- System Calls for Process Management
  - Major system calls

Call	Description
pid = fork()	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

  - Wait, System Calls?

    System Calls are interrupt signals sent by software



    - Is a programmatic way of a program requesting for service to kernel of operating system
    - It's like 'Hey OS, could you do y? It's really important'
    - i.e. Fetching a file in hard-disk drive

- System Calls for File Management
  - Major system calls

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	get a file's status information

- System Call Interface
  - Interface
    - Is a point where two systems, subjects, organizations, etc. meet and interact. (Definition)
  - System Call Interface
    - Is the point where user mode and kernel mode meet and interact

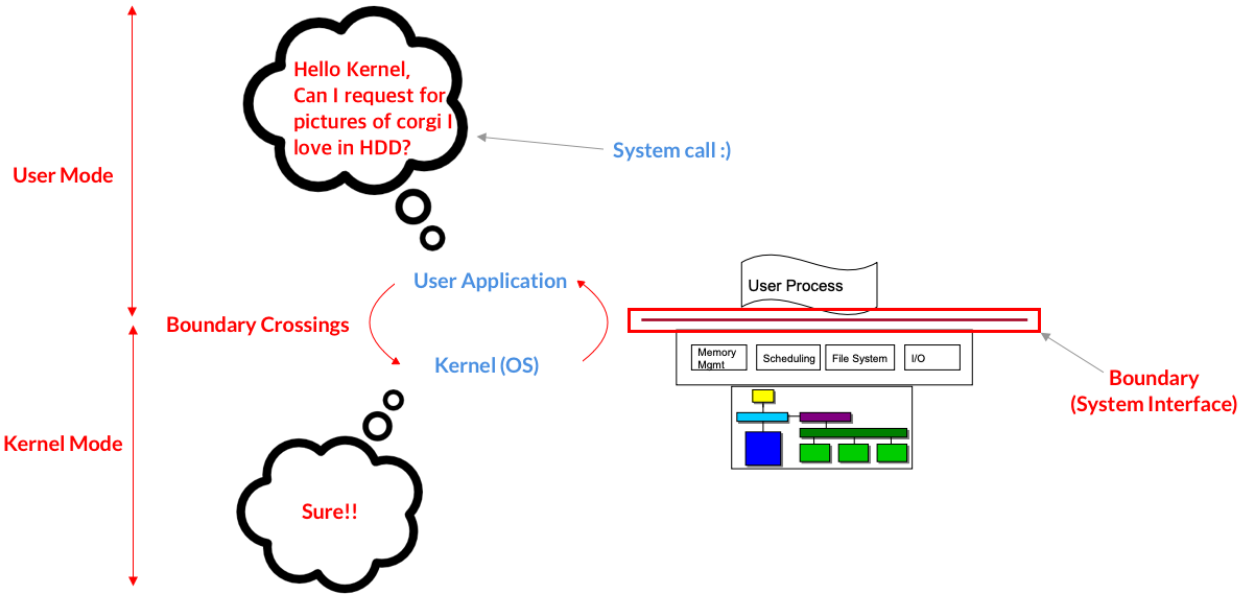
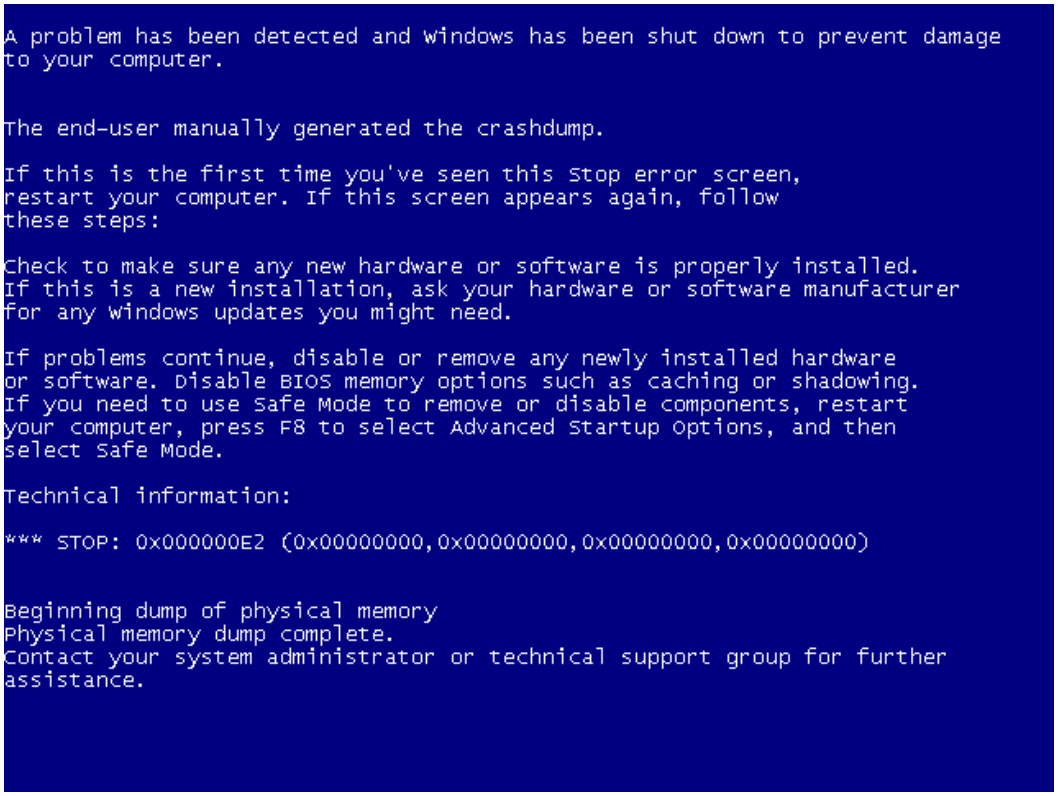


Figure 1: Now, there really is a party going on here :)

    - Most of the system call interface are hidden from the programmer by API
  - User Mode
    - Cannot directly access memory and other hardwares
    - Is safe
    - Crash → doesn't halt entire system
  - Kernel Mode
    - Does have access to memory and other hardwares
    - Is a previleged mode

    - Is not safe
    - Is fragile
    - Crash → halts entire system
        - i.e. The Blue Screen of Death >:)



- System Call Operation
    - Register is a local storage device present in CPU
    - Register may include the address of the memory location instead of real data
    - CPU takes data that has to be processed from register

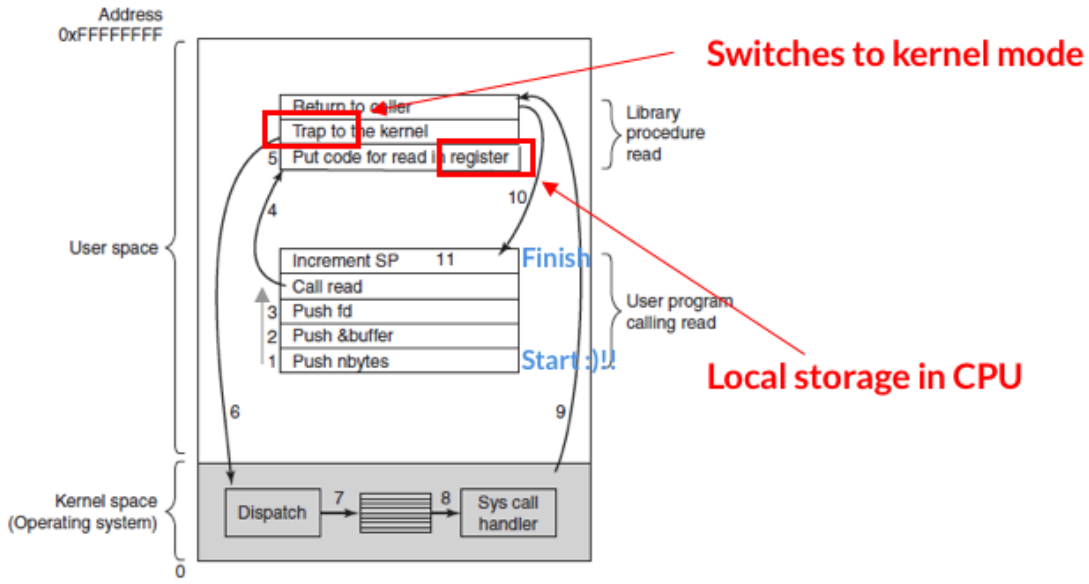


Figure 1-17. The 11 steps in making the system call read(fd, buffer, nbytes).