

# CSc 360

## Operating Systems

### OS Interfaces

Jianping Pan  
Summer 2006

5/8/06

CSc 360

1

## OS services

- User/programmer interfaces
  - command line, GUI, API, system calls
- Program execution
- I/O operation
- File manipulation
- Process communication
- Error handling: software/hardware error

5/8/06

CSc 360

2

## More OS services

- Resource allocation and arbitration
  - CPU, memory, storage, I/O
- Resource sharing and protection
  - among processes, users, computers
  - authentication, authorization, accounting
- Different interfaces to these services
  - regular user, application programmer, system programmer, system designer

5/8/06

CSc 360

3

## Command line interface

- E.g.
  - Microsoft DOS: \command.com
  - Linux: /bin/bash
- Interactivity: interpreter
- Implementation
  - internal: dir (DOS), cd (DOS/Unix)
  - external: ls (Unix)
- Programmability: shell script

5/8/06

CSc 360

4

## Graphics user interface

- E.g.
  - Microsoft Windows
  - K Desktop Environment (KDE)
- Interactivity: point-and-click, drag-and-drop
- Implementation
  - integrated with OS
  - OS front-end
- Programmability: e.g., AutoIt

5/8/06

CSc 360

5

## System calls

- Primitive interfaces to OS services
- System call categories
  - process control
    - fork, exec\*, wait, kill, signal, exit, etc
  - file/device manipulation
    - creat[e], open, read, write, lseek, close, etc
    - socket, bind, listen, accept, connect, etc
  - information manipulation
    - time, getpid, getgid, gethostname, etc

5/8/06

CSc 360

6

## System call examples

- Copy (the content of) file A to file B
  - in CLI: `cp /path/to/a /path/to/b`
  - in GUI: Ctrl-C and Ctrl-V, Ctrl-Drag
- With system calls
  - `open("/path/to/a", O_RDONLY);`
  - `creat("/path/to/b", S_IRWXU);`
    - `open()` with `O_CREAT|O_WRONLY|O_TRUNC`
  - `read()` and `write()`
  - `close()`

5/8/06

CSc 360

7

## System call implementation

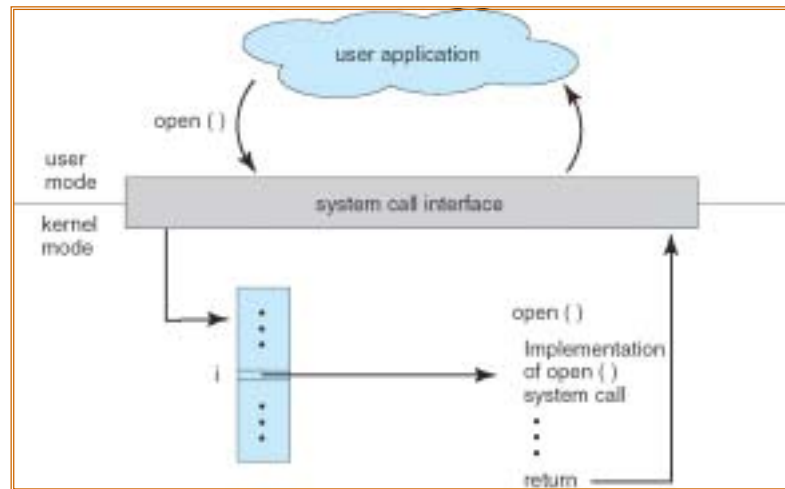
- Software interrupt
  - e.g., INT21H in DOS
  - command: AH (e.g., 2A/2B: get/set system date)
  - parameters
    - in registers
    - on system stack
    - in memory (pointed by registers)
  - return status: in specific registers
  - return data

5/8/06

CSc 360

8

## System call flows



5/8/06

CSc 360

9

## App programming interface

- E.g.
  - Win32 API: Windows
  - POSIX API: Unix, Linux, OSX, (Windows)
  - Java API: Java JVM
- API: another layer of abstraction
  - mostly OS-independent
  - higher level of functionality
    - implemented by a series of system calls and more

5/8/06

CSc 360

10

## API examples

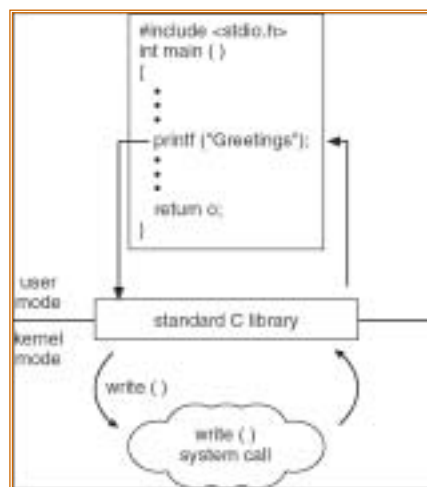
- Copy (the content of) file A to file B
- With C library
  - `fopen("/path/to/a", "r");`
  - `fopen("/path/to/b", "w");`
  - `fread()` and `fwrite()`
    - formatted I/O: element size, # of elements
    - buffered I/O: streams
  - `fclose()`

5/8/06

CSc 360

11

## API flows



5/8/06

CSc 360

12

# Unix manual

- Manual sections
  - 1 user commands
  - 2 system calls
  - 3 C library functions
  - 4 device and network interfaces
  - ...
- E.g.
  - man 1 open; man 2 open

5/8/06

CSc 360

13

# This lecture

- Interfaces to OS services
  - CLI, GUI
  - system calls
  - API
- Explore further
  - compare different OS interfaces for one of your favorite tasks using home computer
  - how to copy file attributes?

5/8/06

CSc 360

14

## Next lecture

- Structures of OS
  - layered, micro-kernel, modular
  - read OSC7 Chapter 2 (or OSC6 Chapter 3)