Interfaces to OS Services

Chapter 2.1, 2.2, 2.3, 2.4, 2.5

Agenda

1. OS Services
2. Command Line Interfaces
3. GUI Interfaces
4. System Calls
5. API
6. Unix Manual

1. OS services (1)

User/programmer interfaces

- command line, GUI, API, system calls

Program execution

I/O operation

File manipulation

Process communication

Error handling: software/hardware error

1. OS services (2)

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

2. 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

3. 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

4. System calls (1)

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

4. System call examples (2)

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

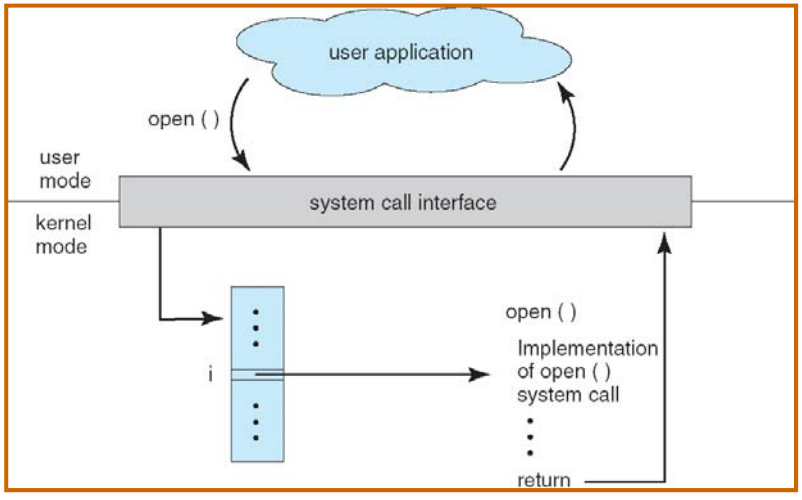
- open("/path/to/a", O_RDONLY);
- creat("/path/to/b", S_IRWXU);
  - open() with O_CREAT|O_WRONLY|O_TRUNC
- read() and write()
- close()

4. System call implementation (3)

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

4. System call flows (4)



5. API (1)

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

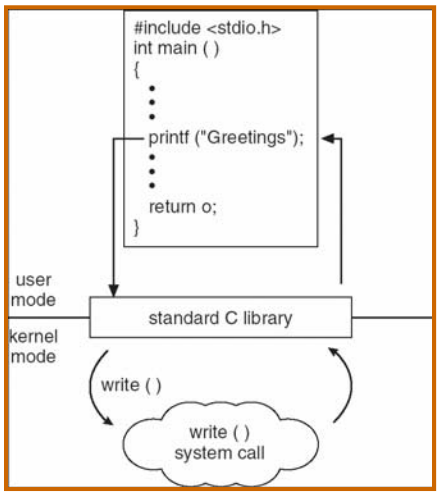
5. API (2): examples

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. API (3): flows



6. Unix manual

Manual sections

- 1user commands
- 2system calls
- 3C library functions
- 4device and network interfaces
- …

E.g.

- man 1 open; man 2 open