Date No

Operating System Services

SOS lec 3

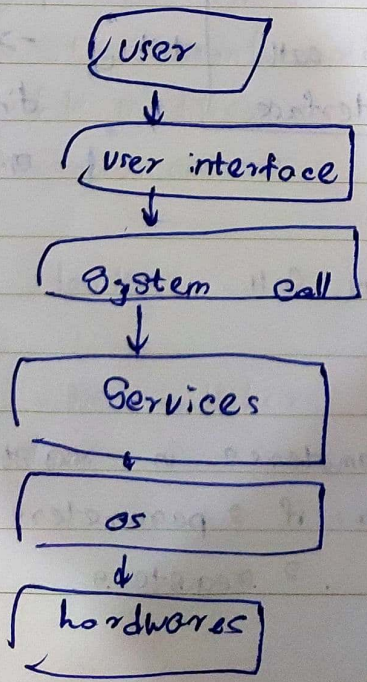
- provide an environment for execute program.
- provide environment for users.

OS provids.

01) User Interface (CLI , GUI)
02) Program execution
03) I/o operation
04) file system manipulation (read, write, create)
05) Communication ove the Network
06) Error detection.
07) Resource allocation
08) Logging
09) Protection & Security.

} OS provids for system it self.

view of OS Service.



System calls.

- how users making requests from the system use will call API and API will call system call.

- System execution mod.
  - User mode (doesn't have directe acces)
  - kernel mode (have direct access to hardwares)

API - Application Programmir

- System call is a programattic way in which a Computer program request a service from the kernel of OS.

Application / user → API → System call → Service.

- each system call have a number.

System Call → System call interface. table → go to location according to call number and return the value.

Passing Parameter from System Call.

3 main methods.

1. Passing parameteres in registers
   - ex: if 3 parameters there will be 3 registers.

Date No

02) Parameters stored in a table and the address of the table pass as a parameter to registers.

03) Parameters moved in to own program stack and pull out in wanted times.

Types of System calls

- Process Control
  Create, terminate, end, abort
- File managment
  Create, delete, open, close.
- Device managment
  read, write, get, set

System Services for program devlopment & execution.

01) file managment
02) Status information
03) File modification
04) Communicotion.

Linkers & loaders

- need linkers & loaders for program execution, and for loading.

Linker - Will link all other object files to the Compiled object file and create the executable file.

Loader - Will load the executable file that created by Linker to the main memory.

- if we create a executable file in one OS and execute in another type of ~~program~~ OS,

to over come this issue program can be written python, C language, Ruby.

- Compile in each and every OS.

Design & implimentation of OS.

- Choosing the type of System
- User goals and System goals.
- Separate policy from mechanisms

policy - What need to be done

mechanism - How to do Something.

- there days using high level language to write an OS.

Date No

OS Structures.

- Simple Structure (MS DOS)
- More Complex Structure (UNIX)
- Layered Structure
- Mickro kernel.

- More Complex Structure (UNIX)

have 2 Separate parts.

1. System programs
2. kernel

- most of functionalties putted into kernel. if you need to change any thing have to change kernel.

- Layered opperacl

- The system has been devided into seperate layers. each layer has a specific function. Lower layer hard ware. higher layer interface.

advantages

- easy to manage
- less coding
- debug can done to seperate layers

- Mickro kernel Structur.

  - Size of the kernel is reduced. removed Some parts from kernel and implimented as user level or system program.

Advantages:

add new features.
more reliable
Secure.
if system fails whole system Wont crash.

Modules.

- Similar to layer but more flexible.
- uses oop.
- additional services linked to the core component.

Hybrid System.

- Combined multiple approaches together to have extra,

Security
Speed.
reliability,
Usability

Date No

Building & booting OS.

Write Source code

↓

Configure os

↓

Compile os

↓

Install os

↓

Boot the OS.

System boot.

- first program running when power the system.
- BIOS loads to main memory and start.