1.1

The system programming is a program that helps a user to execute a program in a computer system
↳ System Programming
↳ Application Programming

Assemblers
- Assembler is a program
- Assembler converts the assembly language to a machine code.
- It ~~conver~~ takes the operation from the assembler & convert them into a machine code which can be identify by some kind of processor.
- An assembler helps the application developer to use, operate & change in the hardware.
- An assembler acts like a compiler & an interpreter for the assembly language.

Linkers
- The tool that merges all the object files together into one executable file
- It is also linker head.

Loader.
- It loads the executable file from the secondary memory to the main memory (RAM)
- Loading a program involves:
  ↳ reading the content of the executable file
  ↳ do other ~~prepatory~~ prepatatory task.
  ↳ pass the control to the executable file when it goes from the secondary memory to the main memory.

Macro processor.

↳ As the name suggest it process the source code.

↳ There are 4 types of processor
- macros
- file inclusion
- other derative
- conditional compilation.

↳ Whenever a compiler sees a macro it converts into actual piece of a code

Compiler	Interpretor
↳ It scans the entire code first d convert them into machine code	↳ It scans the code line by line d then convert into machine code.
↳ Debugging is slow	↳ Debugging is fast.
↳ Compiler show all the error at once	↳ Interpretor show error line by line.
↳ execution time is less	↳ execution time is more
↳ C, C++	↳ Java Python.

Device Drivers.

↳ ~~This~~ When we need a hardware we go to device drivers.

↳ ~~It~~ ~~provides~~ The smooth operation of the hardware is done by device drivers

↳ Any communication ~~of~~ OS needs for hardware

Operating System.

- It's the most important software in the computer system
- It acts as an intermediate between the hardware & software
- Operating system is the collection of the software that controls the hardware fore
  - File Management
  - Scheduling
  - Storage management
  - Security
  - Memory management
- Operating system needs a processor for processing a file / executing a file.
- I/O device / CPU → to store the data
- Input/output devices.

Why

It makes it more efficient, conveince, resource manager.

1.2

The evolution of operating System.

I Early System
- This system only process one job at a time
- Job were collected, and processed one by one without any user interaction.

II Simple Batch System
- early form of OS
- It could take multiple job & process in batches with minimal user interaction.

III Single user, single tasking

IV Single user, multitasking
early version of mac OS

V Multiprogramming.
This help multiple program to be loaded in the memory & executed by the CPU.

VI Mutiple ~~program~~ user, multiprogramming
UNIX

VII Single ~~pro~~ user, multiprogramming
GUIS, computers etc.

VIII network operating system
- enable file sharing, print sharing, communication sharing all over the networks.

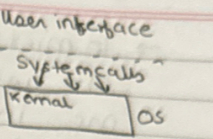
IX ### Modern operating System
- It help us multiprogramming, multi user.
eg: windows, Macs.

X Moblie operating system
- A touch interface device which help to communicate throughout the world

XI Cloud & Virtualization.

1.3 + 1.4

Types of Architecture in OS



I Monolithic
- each component of the OS is contained in the kernal.
- The component call each other using function calls.
eg: Linux

II Layered
- The OS is arranged in layers.
- Layer 0 has hardware the highest layer has user programs
- It is arranged in hierarchical manner so the top layer uses the capibilities of the lower layer.

→ Layers → Hardware, CPU scheduling, memory management, process Management, Input & output, user program

III MicroKernal

MicroKernal	Monolithic
→ The size of the micro-kernal is very small	→ The size of monolithic is more than micro kernal.
→ It's ~~easy~~ difficult to setup the OS for microkernal	→ ~~difficult~~ easy to setup.
→ More coding is required	→ less codding is required
→ Easier to add functionalities	→ Difficult to add
→ execution speed is low	→ ↑
→ Debugging is simple	→ ↑
→ Simple to maintain	→ More resources is required to maintain.

→ If one component fails, the system won't get affected →

→ In microkernal, the user & kernal one in different address unit.

→ In monolithic the components of OS are contained in the kernal.

IV Hybrid
- It's the mixture of many structure.
- Mixture of monolithic & modular components
- Can use layers, Microkernal to optimise the performance ↑ efficiency

1.5
System Calls
↳ It provides the interface between the process & the operating system.
↳ Kernal is the core component of the OS that has control over the hardware.

PDF C1
Process Control → Create a process / terminate a process / load a process / execute / wait / allocate
Device management → request / release / read / write
File management → Create a file / delete a file / open / write / read
Communication → create / delete send & receive message.
Information management → get time, get date, get system data

Process Control
↳ Create a ~~file~~ process
↳ delete a process
↳ execute a process
↳ allocate a process
↳ wait

fork() → It creates a process
exit() → It exits the process
wait() →
Kill() → Terminate
getpid → process id
getppid →

Device management
→ Request a device
→ release a device
→ write
→ read

read ( ) →
write ( ) →
ioctl ( ) → performs I/O operation on the device

File management
read ( ) →
write ( ) →
open ( ) →
close ( ) →
rename ( ) →
link ( ) →

Communication
pipe ( )
MMap ( )

Information
alarm ( ) getpid ( )
sleep ( )

1.7
System Boot

(i) When you turn on the computer, the boot process starts.
(ii) It happens fast, and it begins at the same memory location
(iii) The system firmware runs the first boot instruction.
(iv) The ~~co~~ad loads the program called boot block.
(v) The boot block finds d load the bootstrap program on the disk.
(vi) It locates the OS Kernal loads into the main memory
(vii) A common boot loader is called grub, it help you to choose from different OS if its installed.

Cold booting - Turning on the system from completely off state.

Warm - Restarting the system.

POST [Power on self test]
↓ (The system checks the hardware)

BIOS
↓ This firmware controls communication between OS & hardware

Boot Loader
↓ The BIOS finds the boot loader.

Kernal initialization
↓ The boot loader loads the OS Kernal.

init Process
↓ The kernal start the process called init

User login
↳ It shows the login page.