

WITHOUT USER INTERFACE

S.MOHAN

ASSISTANT PROFESSOR OF COMPUTER SCIENCE

**V.S.B. COLLEGE OF ENGINEERING TECHNICAL CAMPUS,
KINATHUKADAVU –COIMBATORE –TAMIL NADU-642109.**

ABSTRACT :

an operating system without user interface (UI) is a background system that manage the computer hardware and runs hidden task automatically without needing a oerson to look at a server or press bupaten . auser interface is a visual tool – like screen , menu or icon that lets human task to a machine when an operating system has no UI (often called a headless or embedded system) it communicate only with other machines, sensor or internal microchips even without a human interface the operating system still performs its main abstract jobs as a manager . it divide up power from the central processing unit (CPU) and manage for different task . it move the signal from physical sensor to internal programs . it follows the coded instruction instantly when an event happen without waiting for human permission why system use no UI the compute does not want every running graphics or screen machine run the same steps every time without human mistakes communication go straight to the hardware with no middle step.

Keywords : UI, GUI,CLI,CPU,OS

INTRODUCTION :

An operating system (OS) without a user interface or more precisely are without an end-user interactive interface is a specialized system designed to run automatically in the background without human interaction UI- less person means an OS that lacks human facing instruction layer like graphical user interface (GUI) or a commands like interface (CLI) instead of talking direct input from a user via keyboard , mouse or screen it directly manage hardware and executes dedicated , pre-programmed task or machine – to – machine instruction . it relies entirely on the kernel to handle hardware resources , memory allocation and background daemons or automatically triggers.

It is visual system like aircraft auto pilot mechanism –anti lock breaking system (ABS) in a car and industrial robots . they process data and react to sensors input with in strict ore-determined time limits with no end user interface . the found in micro wave oven , digital watches and smart home , micro controller when the software operate purely an internal triggers and physical button then an OS shell.

Which network administrator via SSH (which provide CLI) the base system running core micro service to IoT edge computing can operate entirely headless and automated

OPERATING SYSTEM MEANS :

An operating system (OS) is the core software that manage computer physical hardware and allows different application to run safely and smoothly it save as a bridge between you (the user and the complex machinery inside your device) the operating system co –ordinate the processor (CPU) and memory (RAM) , storage devices like keyboard and monitors . it gives programs a stable platforms to launch and operate without crashing in to each others . it keeps track of your documents , photos, and folders , so you can find it easily it lets you instruct with the computer using visual icon (GUI) instead of raw binary code

MOBILE OPERATING SYSTEM :

an mobile operating system is a specialized software platforms that manage hardware and run application on smart phone , tablets , wearable and other portable devices.

ANDROID :

Developed by google this open-source, linux-based OS power the vast majority of the globe smart phone , offering deep configuration and wide variety of hardware choices

iOS : created by apple exclusively for the iPhone this closed-source software is know for smooth performance high security and tight integration with in the apple device eco-system.

HARMONY OS :

developed by HUAWEI as an alteration platforms designed to function across the multiple smart devices categorized and eco-system.

KAI OS :

a light weight operating system built ti bring essential internet –services and Apps to affordable non-touch feature.

APPLICATION :

App management : code , runs and closes mobile application efficiently .

HARDWARE CONTROLS :

Manage touch screen and cameras, blue tooth Wi-Fi and GPS sensors.

NETWORK

CONNECTIVITY

:

handles cellular voice call text messages and mobile data connection

POWER MANAGERMENTS :

optimization battery consumption to executed device usage time.

OPERATING SYSTEM FUNCTIONS :

hardware managements : co-ordinate the CPU , memory storage, devices and input and output peripherals .

Resource allocation :

Assign system memory and processing power along with the running applications.

File organization :

Keep tracks of files , folders and directories an storage devices for easily retrieval.

Security and production :

Safeguard data and system process from unauthorized accesses or conflicts.

MOBILE OS vs DESKTOP OS: A mobile operating system is a light weight an optimized for touch input and battery conservation where as desk top operating system is comprehensive and built for heavy multi-tasking and precise mouse-used- keyboard control.

User INTERFACE input

Mobile OS :

Designed for touch screen using taps swips and pinches.

Exampels : Android , and iOS

Desktop OS :

Built for processing control using physical keyboard and mouse

Exampels :windows macOS and linux.

RESOUCCE MANAGERMENTS :

mobile OS :

Tightly optimized for power efficiency to extent battery life an device with limited hardware it aggressively suspend or closes background task to save memory.

User strict application and boxing categorized App stores (like Google play or Apple App store) to isolate apps and protect user data.

Desktop OS :

focus on raw complexity performance and heavy multi-tasking using higher RAM storage and processing powers.

After looses the isolation allowing software to be down loaded from any website and grant user deeper access to modify file systems.

HARDWARE connectivity :

In mobile OS built to manage cellular data, SIM card, Wi-Fi, blue tooth and cameras .

In desktop is deluged to connect with the diverse extended peripherals like printer extended device and multi-monitor setups.

Operating system ARCHITECTURE :

the operating system architecture is a internal design blue print that defines how an operating systems organize is compute and manage hardware resources for application the hardware physical dreams like the CPU , memory and disk the kernel the central core of the menu with full privilege and controls hardware directly the system calls a secure programs bridge that lets app request services from the kernel shell /use interface . the create layers (command-line or graphical) where user run programs .

OS structure :

Simple structure :

Has no strict boundaries Apps can Access hardware directly . which makes it fast but prove total system crashes (eg MS_DOS)

Monolithic KERNAL architecture :

Puts all core OS services (file system, memory, diverse) in to one single layer kernel space for memory speed (eg Linux)

Layer architecture :

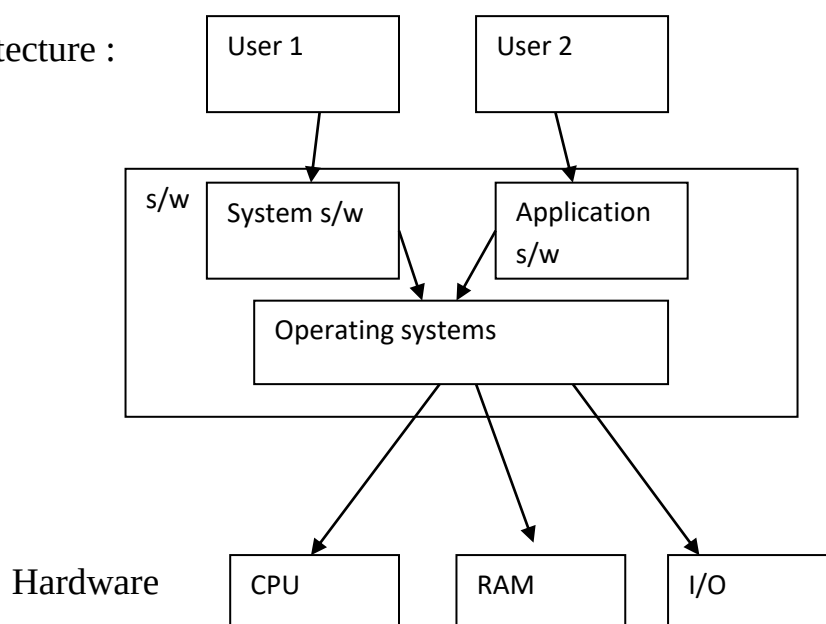
Organize the OS in to stacked functional level , where each layer only uses the service of the layer directly below it making debugging easier(eg THE OS)

Micro kernel architecture :

keeps only absolute core function in the kernel and pushes services like desire and file system in to user space for better security (eg QNX ,MINIX).

HYBRID kernels :
bled monolithic performance with micro kernel modularity but keeping same heavy services inside the kernel space to reduce communication delay (eg windows NT , macOS

Architecture :



WITHOUT USER INTERFECE OS :

An operating system without a user interface (an strictly without a graphical user user interface) means a core system software that manage hardware runs programs entirely through text commands and background operation without visual icon, windows or a mouse.

NO visual elements :

Then an no desktop backgrounds , clickable icon or mouse paths

TEXT based or HEADLESS :
intract layer via a commands line interface (CLI) using typed text commands or it os run completely “DEADLESS” (without a screen or keyboard) in the backgrounds.

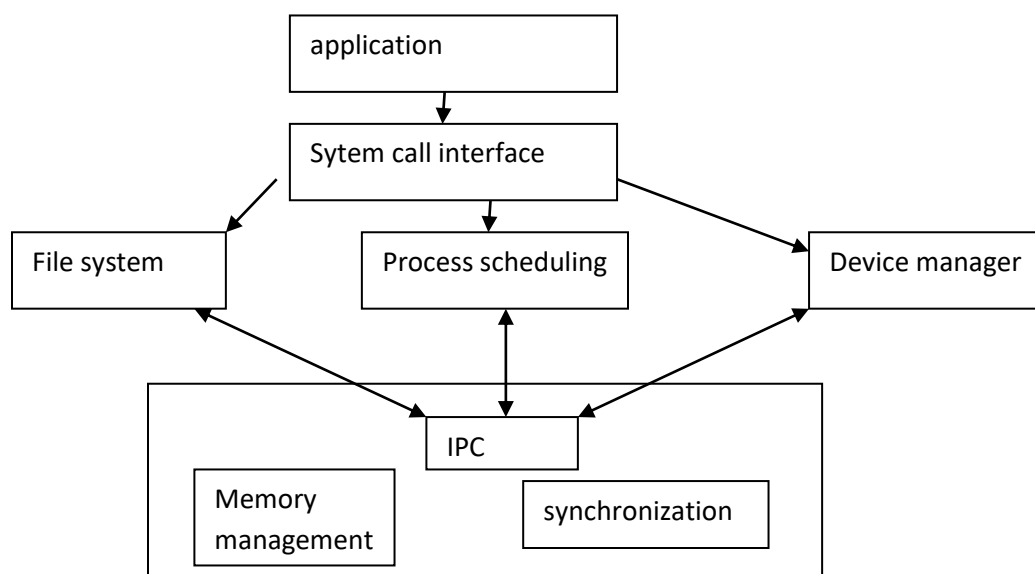
DIRECT RESOURCE controls :
without graphics display draining power and manage , the system user minimal RAM and processing powers .

EMBEDDED systems :

Microchips in house hold application core sensor or medical tools run basic code directly or hardware without a human interface .

SERVER enrichments :

User serve often run Linux without graphics to reserve all processing for hosting web sites or databases.



LEGACY system :
early system like MS-DOS relies completely an typically text commands to open files and programs.

ARCHITECTURE :
an OS architecture without user interface (UI) shift the focus entirely forms human interaction to direct communication between backgrounds applications software , system service and including hardware . a headless or non-UI OS architecture (such as embedded real-time OS or core server environments) is structure vertically6 from application down to physical components headless server controlled script or system daemons that process data handle

networks request or control machinery without rendering visual display or accept direct terminal inputs .

SYSTEM calls AND APIs:

The programmatic interface (like POSIX standard) that lets background software request service from the kernel replacing user- facing shells with pure code level in location.

KERNAL

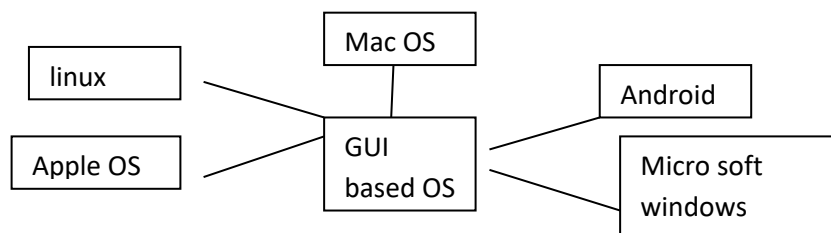
the core engine (whether monolithic kernel or micro kernel) that manages raw CPU scheduling memory location , interfaces communication and device synchronization.

WITHOUT user interface / WITH user interface OS :

An OS with a user interface user uses the visual elements or text commands for human interface , where as an OS without a directly interactive user interfaces (often running heedlessly or in batch./ embedded modes) execute pre-determined task and scripts automatically without a user presents.

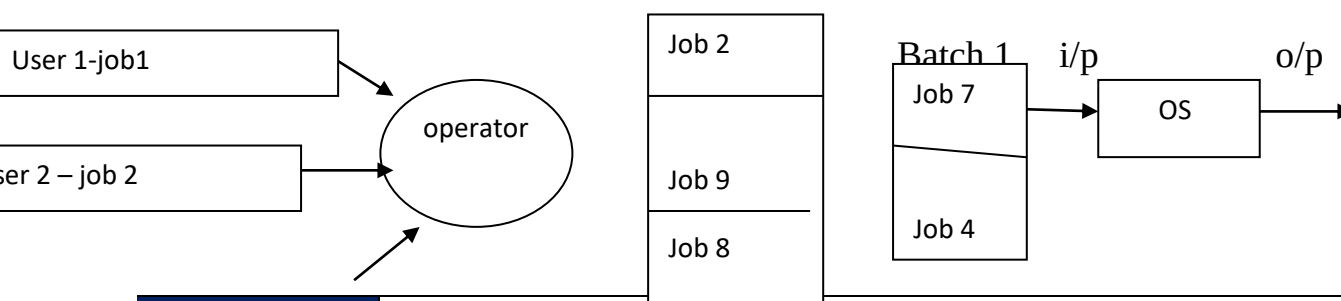
OS with the user INTERFACE :

GUI : allow user to absolutely control hardware and run Apps through visual or text layers. use windows , icon and a mouse (eg windows ,Mac OS)



CLI : user text based text prompt and typed interaction (like Linux bash or windows command prompts)

OS without user INTERFACE : some system are low-level operation model and without a standard human user interface , relying instead of pre-programmed betel job , automated daemons or strict machine-to-machine input.



User n – job n

BATCH process / HEADLESS system :

Extended automated queries of job sequentially without managed real-time .

CONCLUSION :

an OS without a user interface (UI-LESS OS) function entirely in to the background to manage hardware and automated task without human interventions . the UI-Less Os it operate as a pure kernel and resource manager deticately 100% of its processing power and memory to system operate or backgrounds process rather than reducing visuals. These systems are built for embedded devices , IoT sensor router and industrial servers where task run automatically via script or pre-programmed triggers . which they lack human – friendly ease of use , they offer minimum performers , minimal security valurnobility and ultra-low resource consumptions.

REFERENCES :

1. Amazon. *Amzaon High Performance Computing Cloud Using GPU*. Accessed: Jan. 1, 2020. [Online]. Available: <http://aws.amazon.com/hpc/>
Show in Context [Google Scholar](#)
2. P. Carvalho, R. Cruz, L. M. A. Drummond, C. Bentes, E. Clua, E. Cataldo, and L. A. J. Marzulo, “Kernel concurrency opportunities based on GPU benchmarks characterization,” *Cluster Comput.*, vol. 23, no., pp. 177–188, Mar. 2020.
Show in Context [CrossRef](#) [Google Scholar](#)
3. S. Kato, K. Lakshmanan, R. Ishikawa, and Y. Ishikawa, “TimeGraph: GPU scheduling for real-time multi-tasking environments,” in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2011, pp. 17–30.
Show in Context [Google Scholar](#)
4. C. J. Rossbach, J. Currey, M. Silberstein, B. Ray, and E. Witchel, “PTask: Operating system abstractions to manage GPUs as compute devices,” in *Proc. 23rd ACM Symp. Operating Syst. Princ. (SOSP)*, 2011, pp. 233–248.
Show in Context [CrossRef](#) [Google Scholar](#)
5. Zhu, D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and M. Erez, “Kelp: QoS for accelerated machine learning systems,” in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 172–184.

6.R. A. Q. Cruz, C. Bentes, B. Breder, E. Vasconcellos, E. Clua, P. M. C. Carvalho, and L. M. A. Drummond, “Maximizing the GPU resource usage by reordering concurrent kernels submission,” *Concurrency Comput., Pract. Exper.*, vol. 31, no. 18, Sep. 2019, Art. no. e4409.
[CrossRef](#) [Google Scholar](#)

7.S. Najam, J. Ahmed, S. Masood, and C. M. Ahmed, “Run-time resource management controller for power efficiency of GP-GPU architecture,” *IEEE Access*, vol. 7, pp. 25493–25505, 2019.

[Show in Context](#) [View Article](#)
[Google Scholar](#)

8.Z. Lin, H. Dai, M. Mantor, and H. Zhou, “Coordinated CTA combination and bandwidth partitioning for GPU concurrent kernel execution,” *ACM Trans. Archit. Code Optim.*, vol. 16, no. 3, pp. 1–27, Jun. 2019.

9.J. J. K. Park, Y. Park, and S. Mahlke, “Dynamic resource management for efficient utilization of multitasking GPUs,” *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 527–540, 2017.

[Show in Context](#) [CrossRef](#) [Google Scholar](#)

Copyright & License:

© Authors retain the copyright of this article. This work is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.