JUNAID GIRKAR
60004190057
SE COMPS A-3

OPERATING SYSTEMS
ASSIGNMENT - 2

Describe workings of following Operating systems.

1. XV6 OS

Xv6 is a re-implementation of the Unix sixth edition in order to be used as a learning tool. xv6 was developed by MIT as a teaching operating system for their “6.828” course. A vital fact about xv6 is that it contains all the core Unix concepts and has a similar structure to Unix even though it lacks some functionality that you would expect from a modern operating system. This is a lightweight operating system where the time to compile is very low and it also allows remote debugging.

STRUCTURE OF XV6:-

- Monolithic kernel which provides services to running programs called processes.
- Processes use system calls to access kernel services.
- A system call is a procedure call in an OS interface.

- System call enter the kernel space, force the kernel to perform a service and then return
- A process alternates between user and kernel space

SYSTEM CALLS IN XV6 :-

System call	Description
fork()	Create process
exit()	Terminate current process
wait()	Wait for a child process to exit
kill(pid)	Terminate process pid
getpid()	Return current process's id
sleep(n)	Sleep for n seconds
exec(filename, *argv)	Load a file and execute it
sbrk(n)	Grow process's memory by n bytes
open(filename, flags)	Open a file; flags indicate read/write
read(fd, buf, n)	Read n byes from an open file into buf
write(fd, buf, n)	Write n bytes to an open file
close(fd)	Release open file fd
dup(fd)	Duplicate fd
pipe(p)	Create a pipe and return fd's in p
chdir(dirname)	Change the current directory
mkdir(dirname)	Create a new directory
mknod(name, major, minor)	Create a device file
fstat(fd)	Return info about an open file
link(f1, f2)	Create another name (f2) for the file f1
unlink(filename)	Remove a file

PROCESSES AND MEMORY OF XV6:-

- Xv6 time-share processes. Save its CPU registers when the process is not running, and restore them back when it resumes. Process can create another process using fork
- wait returns the pid of the excited child of the current process. wait until one child to exit.
- Parent process and child process are executed with different memory and different registers. Changing a value does not affect another.

- exec system call replaces the calling process' memory with a new memory image loaded from a file. The file is ELF format.
- Xv6 Shell calls getcmd to read a line of user input, then call fork, which creates a copy of the shell process. The parents call wait, while the child is running the command.
- Process wants more memory could call sbrk(n)

FILE OF XV6:-

- The xv6 file system provides data files, which are uninterpreted byte arrays, and directories, which contain named references to data files and other directories.
- Paths that don't begin with / are evaluated relative to the calling process's current directory,
- A file, called an inode, can have multiple names, called links. The link system call creates another file system name referring to the same inode as an existing file.
- The file's inode and the disk space holding its content are only freed when the file's link count is zero and no file descriptors refer to it.
- cd changes current process working directory. When shell runs this cmd, it does not fork child and change dir. Instead, it changes the current working directory of the shell itself. The reason behind is: if forking a child to do cd, we are changing the child process's working directory, not our current shell process.

ADVANTAGES OF XV6:-

- Very small codebase. Being educational, it focuses on how an OS should work instead of implementing everything. A single programmer can easily overview its entire codebase.
- K.I.S.S. Uses clear and simple algorithms only, most sources are no longer than few hundred lines.

- Minimal Assembly. Most of the code is written in ANSI C.
- Uses protected mode. It abstracts the x86 architecture away with minimal lines of code, and it does not rely on BIOS.
- Fairly modern. It supports IOAPIC, LAPIC etc. It does not use obsolete devices such as floppies like other tutorials do.
- Fairly sophisticated. It uses paging for virtual address spaces, and it is a multitasking OS with symmetric multiprocessing (SMP) support.
- Fully functional. Unlike most tutorials and bare-bones, Xv6 really implements everything that has to be implemented in a working kernel.
- Easy to understand. Has a great commentary book, that also points out which parts of the kernel should be refactored for a real-life OS and how.

DISADVANTAGES OF XV6:-

- Unlike Minix, Xv6 is a monolithic kernel, which means OS layers are not separated.
- Source is not separated either, furthermore kernel space code is mixed with user space code. Thankfully there's a list of files grouped by which module they belong. Everything not listed there is user space code.

2. Real Time OS (RTOS)

RTOS is an Operating System that is used in real-time applications to obtain real-time output without buffer delay. To do multiple tasks without compromising on the synchronization in a short span of time is achieved by the *Real Time Operating System (RTOS)*.

In an RTOS, the processing time requirements are calculated in tenths of seconds or shorter increments of time. Real Time Operating Systems uses real-time constraints such as power time and effective utilization of memory. RTOS is a priority based operating system designed to serve real time applications. RTOS requires fewer resources in-order to provide accuracy of the task. It occupies less memory.

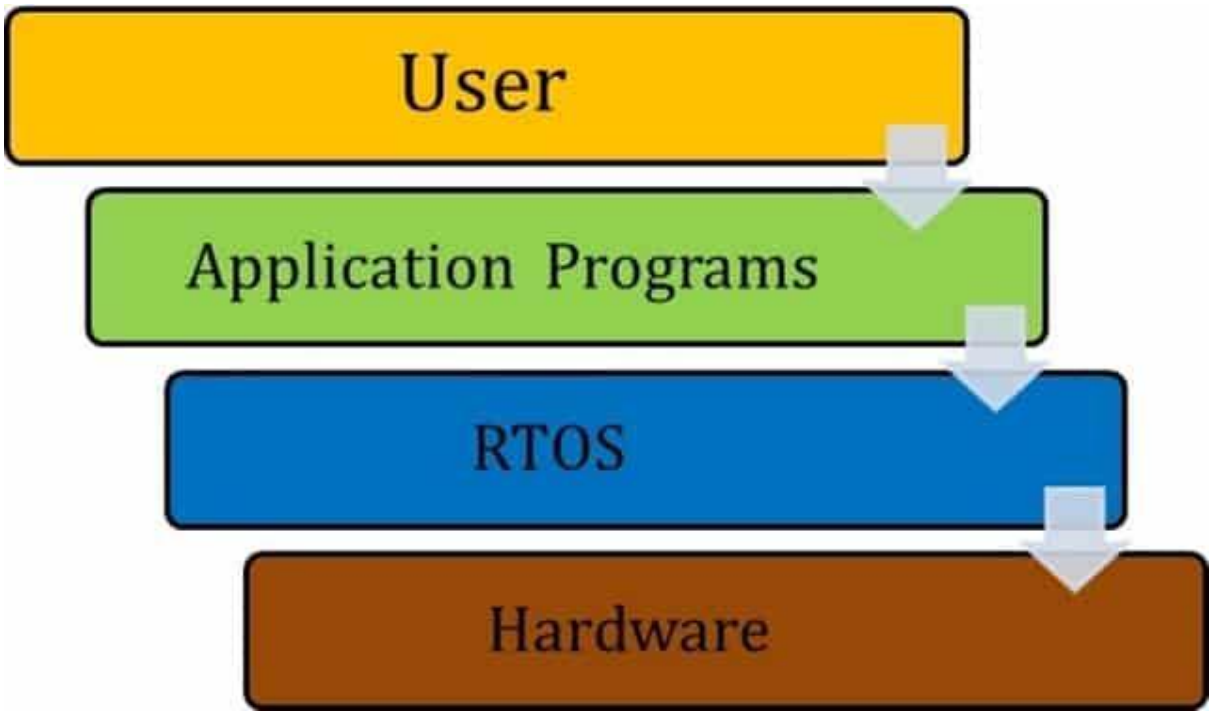

Fig. 2 – Real Time System with RTOS

The General Purpose Operating System is used in PCs and laptops whereas Real Time Operating System finds its

application in embedded systems. Performance is a prominent factor required to be considered while selecting RTOS. The main advantage of a Real Time Operating System is it produces an accurate output within no time. The only disadvantage of RTOS is that the system only concentrates on a few tasks.

Types of RTOS (Real Time Operating System)

It is classified into three types. They are:

1. Soft Real Time Operating System
2. Hard Real Time Operating System
3. Firm Real Time Operating System

1. Soft Real Time Operating System

In this type of Operating System, the response time of the system is prime but not critical to the operation of the system. It has a deadline specified but the system can accept a short amount of delay. Example: Online transaction system, Price quotation system, etc.

2. Hard Real Time Operating System

In Hard Real Time Operating Systems the deadline and the time duration to execute tasks are specified. It is necessary for a system to respond within the time line specified else might

result in disastrous consequences. Example: Medical critical care systems, Aircraft systems, etc.

3. Firm Real Time Operating System

In Firm RTOS, the deadline is specified but missing it does not cause a big impact. *Example*: Multimedia applications.

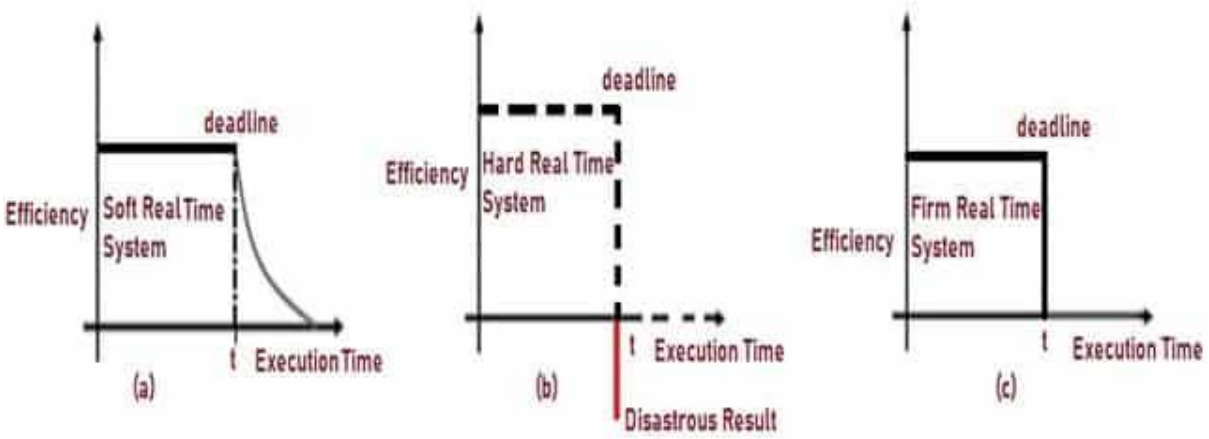

Fig. 3 – Time Efficiency Curves of different RTOS

Parameters for Selection

RTOS is used in Embedded Systems as it requires real-time data. For a Real Time OS to be functional, following parameters are to be considered:

- *Performance*: Performance is the most important factor required to be considered while selecting for an RTOS.
- *Error-Free*: An error-free RTOS performs an error-free task.
- *Maximum Utilization*: Maximum Utilization of the processor can be achieved with the assistance of the RTOS system.
- *Task Shift*: The time taken from shifting one task to another is less in RTOS.

- *Middleware Support*: Middleware support helps Real Time Operating Systems to reduce the time-taken integration of the process.

Relevance of Kernel in RTOS Architecture

The core component of an operating system is called *Kernel*. Micro-Kernel Architecture is implemented in Real Time Operating System with configurable functionalities. Abstraction Layer is provided by the Kernel which offers six main types of common services to the Application software. They are:

- Task Management
- Task Scheduling
- Task Synchronization
- Memory Management
- Time Management
- Interrupt Handling
- Device I/O Management

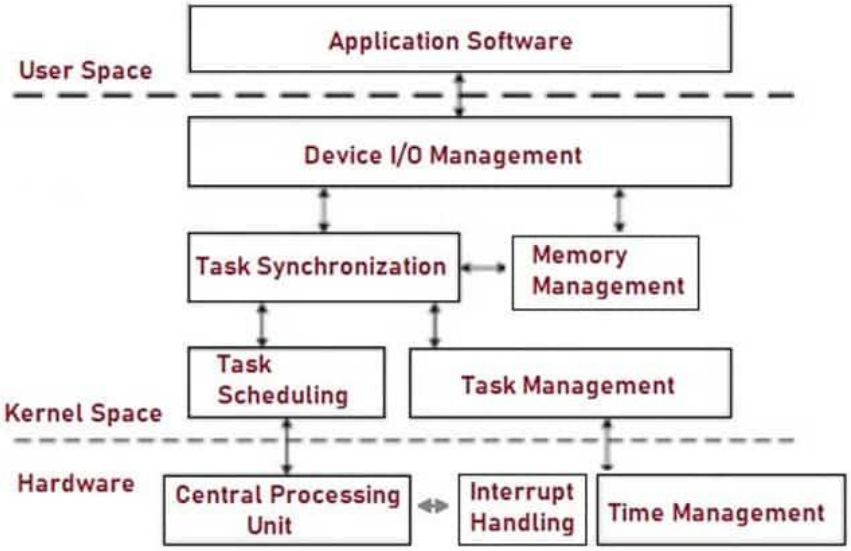

Fig. 4 – Architecture of RTOS

Task Management

The application is divided into small, schedulable, and sequential program units known as '*Thread*' or '*Task*'. This is done to achieve concurrency in Real Time Application. Task Management by Kernel includes Real Time Task Creation, termination, changing priorities etc. Task creation involves creating a *Task Control Block* (TCB) which has information about Task id, priority, Task states i.e. if the task is in (idle, running, ready, terminated) state etc.

Task Scheduling

It records the state of each task and determines the task of highest priority, to be executed. The task which is already running is suspended and the processor executes the high priority task.

Task Synchronization

It is necessary for the information to be transmitted safely from one Task or Thread to another Task. Task Synchronization enables the tasks to mutually share the resources like buffers, I/O devices etc.

Memory management

It allocates the memory for each program. There are two types of Memory Management for RTOS. They are:

- Stack Management
- Heap Management

Time Management

To schedule the tasks that need to be executed during specified duration, there is need for a periodical interrupt. Hence hardware Timer is programmed to interrupt the processor. Time interrupt is called *System Tick*.

Interrupt Handling

CPU is informed about any asynchronous event through an *Interrupt*. It is a hardware mechanism which handles an event by providing functions like defining Interrupt Handler, creation and deletion of Interrupt service routine etc.

Device I/O Management

Device I/O Management helps in providing uniform framework (API – Application Programmers Interface). It also helps in accessing specific hardware device drivers i.e. it locates the right device for I/O request.

Working of RTOS

Real Time Application requests are serviced by Real Time Operating System. The RTOS allows multiple tasks or programs to execute simultaneously based on its priority. Task scheduling Unit decides which thread is to be executed. The processor suspends the running task (if any) and executes the high priority task it receives.

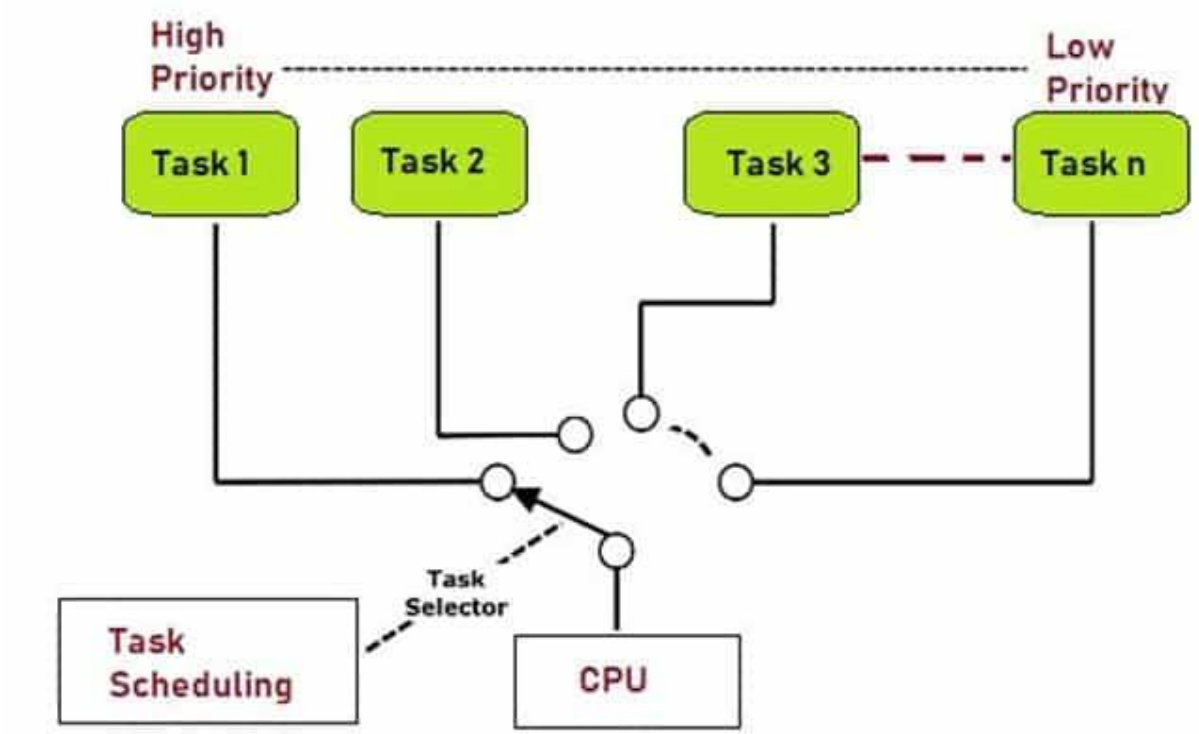

Fig. 5 – Schematic Representation of Working of RTOS

Let us say, a user is browsing on the net and after few seconds switches on YouTube Video and in no time, the user starts listening to a Podcast. Browsing on the net is considered as ‘Task 1’, Video on YouTube is Task 2 and listening to Podcast is assumed to be Task 3. Switching between these tasks is called as Multi-Tasking and RTOS provides efficient multitasking services.

When the User shifts to Task 2, then Task 1 is terminated and executes Task 2 as its priority is High. Similarly, when the User shifts to Task 3, then Task 2 gets terminated and task 3 is executed. Task Scheduling Unit takes care of these functions. Inter task communication, Synchronization, Time management is taken care of by RTOS Kernel. Schematic representation of the same is shown in the fig. 5 above.

Uses of RTOS

RTOS finds its application in embedded systems because of its accurate real-time data. Some of the major applications are listed below:

- Mobile applications.
- Online transaction system.
- Medical Critical Care systems.
- Aircraft and defense systems.
- Price quotation systems.
- Network and multimedia systems.
- Anti-Lock brake systems.

- Air traffic control systems.
- Online calling.
- Ticket reservation systems.
- Command control systems.

Advantages of RTOS

The advantages of Real-Time Operating System include:

- RTOS is event-driven with no processing time delay.
- Real Time OS offers task-based API development. This helps designers or testers to work independently on their parts of the project.
- It reduces the interdependencies between the modules by abstracting timing dependencies and task-based design.
- It offers cleaners and smaller application courses.
- Priority-based scheduling allows the user to separate analytical processing time and Critical processing time.

Disadvantages of RTOS

The disadvantages of RTOS include:

- RTOS requires specific drivers for faster response time.
- It requires plenty of resources, making it more expensive.
- It uses a complex algorithm, which makes it very difficult to interpret.

- It concentrates only on the accuracy of the program under execution, which increases the waiting time of low priority programs.
- RTOS carries out only the minimum switching of tasks.
- RTOS can only run minimal tasks together.
- It uses plenty of resources, which sometimes are not suitable for the system.
- It concentrates only on applications containing an error.

3. Mobile OS

A mobile operating system is the software that makes all these activities possible.The mobile operating system is always running in the background, managing hardware and processing tasks so that your programs work together in a coordinated manner. In addition to apps, it also manages your smartphone features, including multimedia controls, internet connectivity, cameras, GPS navigation and security.

FEATURES AND ITS WORKING :-

Internet connectivity

Mobile operating systems differ from desktop operating systems in one major way. They're able to connect to the internet directly via the mobile phone network. They do this by using a wireless service provider and the smartphone's built-in modem. By comparison, a desktop OS relies on an external Ethernet connection or home broadband network connection for access.

Native web browser

Another typical feature is a native web browser application, which is built into the platform to allow users to access web pages from the home screen. Typically this is Google Chrome on an Android device and Safari on an iPhone but there are other web browsers available to download and use from your app store of choice.

GPS features

Your smartphone probably also came loaded with a GPS application that uses the device's hardware to track your phone's position. It's the mobile operating system that provides the link between that hardware and the GPS-based app you see on your screen. It also allows you to share your location with other devices and updates it as you move.

App stores

Fresh-out-of-the-box phones also include native email, calendar and contacts apps, all running on your smartphone's OS. Mobile operating systems make it easy to find and download more apps for your phone, with the inclusion of a preloaded app store on the home screen. This gives you access to apps that aren't included on your phone by default.

Popular platforms of the Mobile OS

1. Android OS: The Android operating system is the most popular operating system today. It is a mobile OS based on the Linux Kernel and open-source software. The android operating system was developed by Google. The first Android device was launched in 2008.

2. Bada (Samsung Electronics): Bada is a Samsung mobile operating system that was launched in 2010. The Samsung wave was the first mobile to use the bada operating system. The bada operating system offers many mobile features, such as 3-D graphics, application installation, and multipoint-touch.

3. BlackBerry OS: The BlackBerry operating system is a mobile operating system developed by Research In Motion (RIM). This operating system was designed specifically for BlackBerry handheld devices. This operating system is beneficial for the corporate users because it provides synchronization with Microsoft Exchange, Novell GroupWise email, Lotus Domino, and other business software when used with the BlackBerry Enterprise Server.

4. iPhone OS / iOS: The iOS was developed by the Apple inc for the use on its device. The iOS operating system is the most popular operating system today. It is a very secure operating system. The iOS operating system is not available for any other mobiles.

5. Symbian OS: Symbian operating system is a mobile operating system that provides a high-level of integration with communication. The Symbian operating system is based on the java language. It combines middleware of wireless communications and personal information management (PIM) functionality. The Symbian operating system was developed by Symbian Ltd in 1998 for the use of mobile phones. Nokia was the first company to release Symbian OS on its mobile phone at that time.

6. Windows Mobile OS: The window mobile OS is a mobile operating system that was developed by Microsoft. It was designed for the pocket PCs and smart mobiles.

7. Harmony OS: The harmony operating system is the latest mobile operating system that was developed by Huawei for the use of its devices. It is designed primarily for IoT devices.

8. Palm OS: The palm operating system is a mobile operating system that was developed by Palm Ltd for use on personal digital assistants (PADs). It was introduced in 1996. Palm OS is also known as the Garnet OS.

9. WebOS (Palm/HP): The WebOS is a mobile operating system that was developed by Palm. It based on the Linux Kernel. The HP uses this operating system in its mobile and touchpads.

Issues and Challenges of Mobile OS

- Mobile OS design suffers from usability and interoperability problems. Usability problems are difficult due to the small physical size of the mobile phone form factors. Interoperability issues arise from the platform fragmentation of mobile devices, mobile operating systems and mobile browsers.
- Hardware and software configuration management.
- Content delivery for various smartphones operated by various service providers is difficult
- Introduces the possibility of the configuration errors, bugs (in both the OS itself and in the applications, it runs), viruses and other malware.
- Adaptability of applications on various Mobile OS’s. Designing of an app for more than one mobile OS requires more than one design and every mockup has to be intuitive for the specific groups of users (specific to OS).

- Personalization is considered to be biggest challenge as it is the key enabler for the success of the OS.
- System integrity
- Power management
- Continuous Connectivity
- User Interface designs for various mobile apps.
- Approach for positioning of apps for Navigation by various Mobile OS is different.
- Space management and resource saving elements like pop-overs, alerts, software gradients, graphic elements etc.
- Testing of the applications on a plethora of mobile OS’s.

CONCLUSION:

Operating Systems are the most crucial software in modern electronics. We learnt about the different types of OS, their working of specific tasks, how they differ from each other and the advantages and disadvantages of each.