**Study on Mobile Operating System**

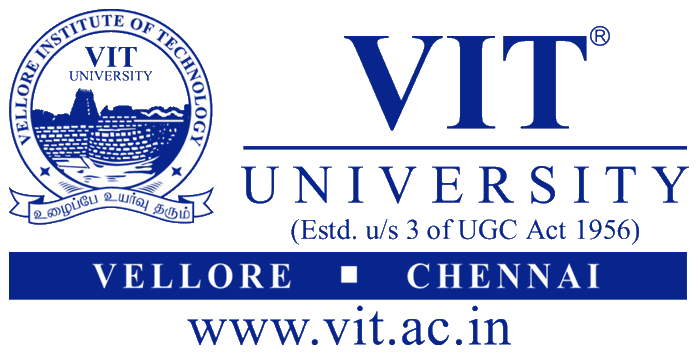
By

Marla Akhil Reddy 13BCE1078

Jagmohan Singh 13BCE1056

Samarth Gupta 13BCE1121

**CSE222 (Operating Systems)** Project report submitted to the **SCHOOL OF COMPUTING SCIENCE and ENGINEERING**



**VIT UNIVERSITY, CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600127.**

**MAY 2015**

Faculty In-charge: **Prof.Jagadeesh Kannan R**

**ABSTRACT**

As we all know that the world is in transition to mobile computing from desktop pc and laptops. There are almost six billion users around the globe. The mobile phones around us are the combination of hardware and the software. The hardware is the hard part of device and the software are the governor of the hardware.Important Mobile Operating systems available are Android, iOS, Windows etc… Android is an open source operating system used use in android mobiles. The iOS is an operating system developed by apple corporation and used in apple products like iPad,iPhone etc. Windows mobile operating system developed by Microsoft itself and embedded by many different mobile companies like Nokia.There are different OS for mobile among them we are going to pick up one of them. Its Android OS.   
In this Project we have looked into important aspects of mobile OS takeing different aspects such as system calls, Memory Managemnt, Process Management, Application management etc...

We developed an application similar to Task Manager which give information about Free memory,Running Process and System Apps.

**TABLE OF CONTENTS**

* **Abstract**
* **Introduction**
* **System Calls**
* **Architecture**
* **Application Life Cycle**
* **Multitasking**
* **Conclusion**
* **References**

**Introduction**

[Operating Systems](http://www.engineersgarage.com/articles/operating-systems-tutorial) have developed a lot in last 15 years. Starting from black and white phones to recent smart phones or mini computers, mobile OS has come far away. Especially for smart phones, Mobile OS has greatly evolved from Palm OS in 1996 to Windows pocket PC in 2000 then to Blackberry OS and Android.

One of the most widely used mobile OS these days is **ANDROID**. **Android** does a software bunch comprise not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been number of updates in the original version of Android.

**Android** is a powerful Operating System supporting a large number of applications in [Smart Phones](http://www.engineersgarage.com/articles/smart-phones). These applications make life more comfortable and advanced for the users. Hardwares that support Android are mainly based on [ARM architecture](http://www.engineersgarage.com/articles/arm-advanced-risc-machines-processors) platform.

Android comes with an Android market which is an online software store. It was developed by Google. It allows Android users to select, and download applications developed by third party developers and use them. There are around 2.0 lack+ games, application and widgets available on the market for users.

Android applications are written in java programming language. Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. For software development, Android provides **Android SDK** (Software development kit).

**Applications**

These are the basics of Android applications:

•      Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers)

•      Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other applications)

•      The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required

•      Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages)

Google, for software development and application development, had launched two competitions ADC1 and ADC2 for the most innovative applications for Android. It offered prizes of USD 10 million combined in ADC1 and 2. ADC1 was launched in January 2008 and ADC 2 was launched in May 2009. These competitions helped Google a lot in making Android better, more user friendly, advanced and interactive.

**System Calls**

**RESTART\_SYSCALL**

restart\_syscall - restart a system call after interruption by a stop signal.

DESCRIPTION

The restart\_syscall() system call is used to restart certain system calls after a process that was stopped by a signal (e.g., SIGSTOP or SIGTSTP) is later resumed after receiving a SIGCONT signal. This system call is designed only for internal use by the kernel.

RETURN VALUE

The return value of restart\_syscall() is the return value of whatever system call is being restarted.

**\_EXIT()**

\_exit, \_Exit - terminate the calling process

DESCRIPTION

The function \_exit() terminates the calling process "immediately".

Any open file descriptors belonging to the process are closed; any children of the process are inherited by process 1, init, and the process's parent is sent a SIGCHLD signal. The function \_Exit() is equivalent to \_exit().

RETURN VALUE

These functions do not return

**READ()**

size\_t read(int fd, void \*buf, size\_t count);

DESCRIPTION

read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.

On files that support seeking, the read operation commences at the current file offset, and the file offset is incremented by the number of bytes read. If the current file offset is at or past the end of file, no bytes ar

RETURN VALUE

On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number.e read, and read() returns zero.

**\_OPEN**

Description:

Given a *pathname* for a file, **open**() returns a file descriptor, a small, nonnegative integer for use in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process. By default, the new file descriptor is set to remain open across an execve. A call to **open**() creates a new *open file description*, an entry in the system-wide table of open files. The open file scription records the file offset and the file status flags (see below). A file descriptor is a reference to an open file description; this reference is unaffected if *pathname* is subsequently removed or modified to refer to a different file.

RETURN VALUE  :

**open**(), **openat**(), and **creat**() return the new file descriptor, or -1 if an error occurred (in which case, *errno* is set appropriately).

\_**CHDIR**

DESCRIPTION  :

**chdir**() changes the current working directory of the calling process to the directory specified in *path*. **fchdir**() is identical to **chdir**(); the only difference is that the directory is given as an open file descriptor.

RETURN VALUE  :

On success, zero is returned. On error, -1 is returned, and *errno* is set appropriately

\_**LSEEK**

DESCRIPTION  :

The **lseek**() function repositions the offset of the open file associated with the file descriptor *fd* to the argument *offset* according to the directive *whence* as follows: **SEEK\_SET** The offset is set to *offset* bytes. **SEEK\_CUR** The offset is set to its current location plus *offset* bytes. **SEEK\_END** The offset is set to the size of the file plus *offset* bytes. The **lseek**() function allows the file offset to be set beyond the end of the file (but this does not change the size of the file). If data is later written at this point, subsequent reads of the data in the gap (a "hole") return null bytes ('\0') until data is actually written into the gap.

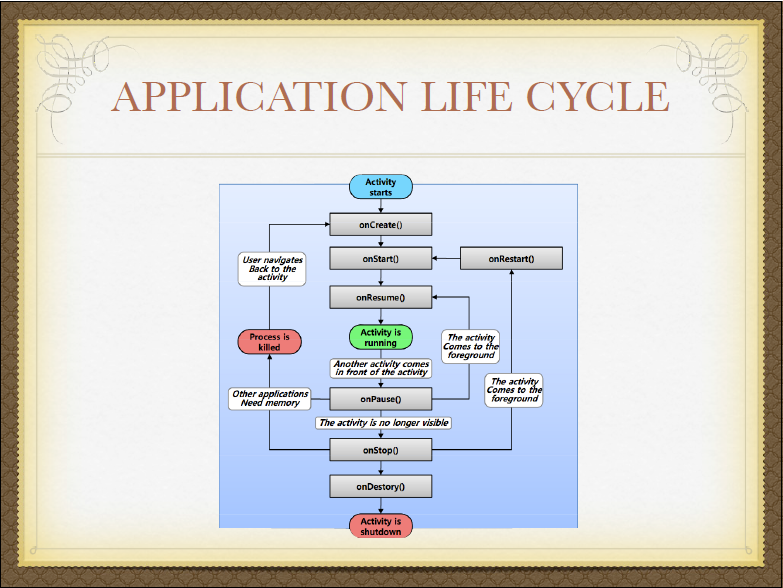
RETURN VALUE  :

Upon successful completion, **lseek**() returns the resulting offset location as measured in bytes from the beginning of the file. On error, the value *(off\_t) -1* is returned and *errno* is set to indicate the error.



**ARCHITECTURE**





**MULTITASKING**

****

REFERENCES

1. <http://syscalls.kernelgrok.com/>
2. <https://developer.android.com/training>