CECS 526

Spring 2020

Cal State Long Beach

Term Paper Assignment

Neha Bhoi, Liam Nguyen

Mobile vs Traditional OS (Case study: Linux vs Android)

Abstract

Operating system (os) is a type of system software that manages computer hardware and provides services for users. Traditional operating system focuses on delivering speed with powerful CPU. However, mobile devices have more constraints than traditional desktops. Mobile phones are less powerful, which supports limited types of user interaction and runs on a set amount of battery power. Thus, mobile os design requires a small footprint and higher efficiency per power usage. Although there are many choices among traditional and mobile os in the market, we choose Android as a representative for mobile os and Linux for traditional os. Linux is the leading operating system on servers and other big iron systems such as mainframe computers, and the only OS used on TOP500 supercomputers [1]. Android is the most popular mobile operating system nowadays both for smartphones and tablets [2]. Android may be based on Linux, but it's not based on the type of Linux system we may use on the desktop. This paper presents a detailed view of differences in hardware architecture, kernel design, underlying file system, power management, and c library support.

Reference:

[1] A. Prakash. Linux runs on all of the top 500 supercomputers, again!, 2020. URLhttps://itsfoss.com/linux-runs-top-supercomputers/. Online.

[2] P. D. P. Bhargavi Padhya, Prasad Desai. Comparison of mobile operating systems.Inter-national Journal of Innovative Research in Computer and Communication Engineering,4(8), 20

1 Introduction - Neha, Liam

The operating system is the most important software that runs on a computer. It manages the computer's memory and processes and coordinates the use of resources efficiently. Generally, OS comes pre-loaded on any computer when we buy them. Most people use the operating system that comes with their computer, but it's possible to upgrade or even change operating systems. The popular operating systems for personal computers are Microsoft Windows, macOS, and Linux. Mobile devices like phones, tablet computers, and MP3 players are different from desktop and laptop computers, therefore they run operating systems that are designed specifically for mobile devices. Some popular examples of mobile operating systems include Apple iOS and Google Android. Mobile operating systems generally aren't as fully featured as those of traditional operating system, and they aren't able to run all of the same software. Therefore, there are many differences between a mobile operating system and a traditional OS. In this paper, we will compare and contrast traditional OS and mobile OS on four aspects: kernel, file system, power management, and security.

2 Target Architecture - Liam

Generally, mobile OS operates on mobile devices which are an embedded system. Since most mobile devices have to work on a battery power source, most of the mobile phone has ARM processors instead of CSIC CPU. ARM and x86 refer to different types of instruction sets. Instruction Set (ISA) provides commands to the processor, to tell it what it needs to do. It usually consists of addressing modes, instructions, native data type, etc... CSIC architecture provides a full set of instructions to provide the max capabilities to the CPU. It utilizes a large, specialized, and complex instruction set. As a result, each instruction can span multiple clock cycles to execute. This pushes the complexity toward hardware, so it is efficient in the use of RAM. CSIC is more used in traditional desktop computers and laptops. However, mobile devices operate on different constraints, so CSIC instruction set is not suitable. Instead, ARM is a smaller instruction set that utilizes a small, highly optimized set of instructions. Its simple instruction only takes one clock cycle to execute, so it pushes the complexity to the compiler. Due to its simplicity, it is also easy to parallelize work with heavy use of RAM. It also has a great performance-per-watt ratio than CSIC. For example, the ARM7100 consumes only 72mW when operating at 14MIPS, 33mW in idle mode, and 33uW during standby. In comparison, the smallest of the Intel's x86 processor, the Atom, needs 1W for operation.

3 Kernel

3.1 What is a kernel? - Neha, Liam

[1] The kernel is a important model of an operating system. It is the part of the operating system that loads first and always remains in the main memory during the runtime of the operating system. As it stays in memory, its footprint has to be small and live in a protected area of the memory. It is in charge of memory management, process management/task

management, and disk management. It acts as a bridge between user space where user applications live and low level operations in CPU and memory. There are a few types of kernels:
Monolithic kernels: The kernel is a single program that operates simply within the privileged mode and in its own address space. Communication between running applications and the kernel side is handled through system calls.
Micro-kernels: The operating system contains a smaller kernel, which operates within the privileged mode and in its own address space. To support the expected range of functions, the operating system is extended by several subsystems referred to as servers. Servers operate with user applications using user mode and the user's section of memory's address space. Communication between application software and kernel is much like with the monolithic kernel, but the kernel calls upon servers to handle many of the operations.
Hybrid kernels: The hybrid kernel compromises between the two extremes where the kernel is still kept small, but server-like components are added on. Here, the server components run in kernel mode but frequently in the user's address space.

3.2 Linux Kernel - Neha

[1]Linux was developed around a monolithic kernel. To advance on the efficiency and lower the impact of errors of the monolithic kernel, the modern monolithic kernel incorporate numerous modules, each of which implements one or more of the core kernel's responsibilities. Modules are loaded either at kernel initialization time or on-demand. Thus, the kernel can itself be modularized. The efficiency of a monolithic kernel's execution is in part depends on the number of modules loaded. Loading little modules leads to a more efficiently executing kernel. Through modules, the Linux kernel can be kept relatively small.

Loading and Removing Modules - Neha

As a system administrator, you can tune the kernel by adding and removing modules. To see the modules that are loaded by default, use the command lsmod. This will give you information like the modules that are loaded, their size, and what software or service might use that module. rmmod and insmod commands will alter the modules that are loaded when you next boot the system; they do not impact your system as it is currently running.

```
Command to remove module : rmmod
Command to add module: insmod
Command to add a module on demand: modprobe
```

3.3 Mobile Kernel - Liam

[2] The GNU C library project provides the core libraries for the GNU system and LINUX system. It was written in C first but now has transitioned to C++. However, this library is licensed under GNU Lesser Public License (LGPL) which restricts derivative works. Also, since GNU library is written with the traditional desktop system in mind, Android OS developers felt that it was too overpowered for mobile devices. An example of this is Native POSIX Thread Library (NPTL). This library provides performant threads that are beneficial

in server applications. However, it's not suitable for phones since phones can only run a small number of concurrent processes. Another example of a wasteful feature is inter-process (IPC). While Android OS still supports IPC which is called "Service", it's limited in features compared to one in Linux. In addition, Android OS supports "Intent Service" which is a service that is activated when a request is active. After the request is handled, the service is stopped. This strategy is implemented to conserve battery power. With these trade-off in mind, Android OS developers forked certain parts of GNU library and combined with their own C library to create a library called "Bionic". In comparison to 400KB in GNU C library, bionic is half that size. [3] While bionic is written in C++, it doesn't include C++ exceptions. The reason for this is because the run time language for Android is Java. Since Java handles all the exceptions in run time, bionic doesn't have to support it. Also, many convenient libraries in C++ STL are removed. The reason for this removal is because STL leads to bloated error messages, an increase in memory and CPU usages, and slow codes. It was estimated that STL libraries are upwards of 100KB.

3.4 Mobile OS stack - Liam

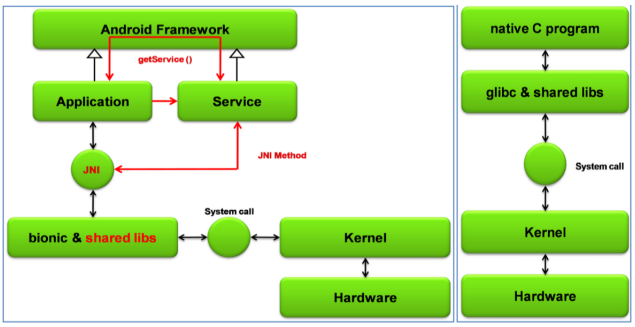

Figure 1: Android vs Linux Architecture [4]

At a high level view, Android operating system has four major components. From high to low abstract level, they are applications, application framework, library, and bionic kernel. Here is the breakdown for each component[5]:

- Application layer: this layer supports user's space applications. They include basic applications such as email client, sms, calendar, maps, etc.... All of these applications are written in Java.
- Application Framework: this layer allows developers to have full access to APIs which are used by their applications. This architecture is designed to simplify the reuse of components so that any application can publish its capabilities and other applications can then use those capabilities as long as they are under the rules of the framework. These are the supported services: views, content providers, resource manager, notification manager, and activity manager.
- Libraries: these are high level libraries written in Java. In this level, every application runs on its process with its instance of Dalvik Virtual Machine. These processes are usually compiled just in time so that they can be run efficiently.

- Kernel: this is the bionic kernel that acts as an intermediate for memory management, process management, network, and drivers.

3.5 Dalvik Virtual Machine - Liam

[4]Besides its custom kernel, Android operating system also has a virtual machine. Although Android kernel is written in C/C++, all other programs are in Java which requires a virtual machine to read bytecodes. As a result, the Dalvik Virtual Machine (DVM) is created. However, unlike other Java Virtual Machine (JVM), DVM uses a special type of bytecodes, so the native Java byte cotes are not executable in Android phones. However, this special type of bytecode is just a more highly optimized version of the native bytecode to perform as efficiently as possible on a slow single CPU with limited memory usage. When phones are first introduced to the market, the hardware is limited. CPU was single and slow, RAM is limited and flash drive space is small. As a result, originally, bionic didn't support OS swapping. OS swapping is a memory technique where OS allocates a certain partition on hard disk for memory called swap space. Then, the OS detects blocks (pages) that haven't been accessed recently. The memory management program then swaps these infrequently used pages of memory out to this swap space and frees up memory. However, due to flask drive constraint in phones, there's no OS swap supported in kernel in Android. Therefore, the amount of virtual memory is governed by the amount of physical memory available on the device. However, recent phones with more powerful hardware have slowly allowed this feature to come back to the mobile OS. Therefore, DVM has been implemented to allow multiple VM executions efficiently. Also, a just-in-time (JIT) compiler has been incorporated to translate Dalvik byte-codes into machine code. In recent Android OS updates, more advanced JIT and garbage collection has been improved and deployed, further these optimizations.

4 Storage Media and File System

4.1 For Linux - Neha

[6]Linux has robust systems and toolings to manage hardware devices, including storage drives. A block device is nothing but the hardware that can be used to store data, like a traditional spinning hard disk drive (HDD), solid-state drive (SSD), flash memory stick, etc. As the kernel interfaces with the hardware by referencing fixed-size blocks, or chunks of space hence it is called a block device because. Disk partitions is nothing but the breaking up a storage drive into smaller usable units.Disks can be formatted and used without partitioning. Some operating systems anticipate to find a partition table, even if there is only a single partition written to the disk. They generally suggested to partition new drives for greater flexibility down the road. While partitioning a disk, we should know what partitioning format will be used. Most operating systems has a choice between MBR (Master Boot Record) and GPT (GUID Partition Table). While the Linux kernel can recognize a raw disk, the drive cannot be used as-is. To use it, it must be formatted. Formatting is the process of writing a file system to the disk and preparing it for file operations. A file system is a system that structures data and controls how information is written to and retrieved

from the underlying disk. Without a file system, you could not use the storage device for any file-related operations.

4.2 File System in Linux - Neha

[7] Some of the more popular file systems for Linux are Ext4, XFS, Btrfs, ZFS. The most popular default file system is Ext4. The Ext4 file system is journaled, backward compatible with legacy systems, incredibly stable, and has mature support and tooling. XFS specializes in performance and large data files. It formats quickly and has good throughput characteristics when handling large files and when working with large disks.XFS has live snap shorting features. XFS uses metadata journaling as opposed to journaling both the metadata and data. Btrfs is a modern, feature-rich copy-on-write file system. This architecture permits for some volume management functionality to be incorporated within the file system layer, together with snapshots, cloning, volumes, etc. ZFS is a copy-on-write file system and volume manager with a robust and mature feature set. It has significant data integrity features, can handle large file system sizes, has typical volume features like snap shorting and cloning, and can organize volumes into RAID and RAID-like arrays for redundancy and performance purposes.

Features		Ext4	XFS	BtrFS	ZFS
Resize Capabilities	Online Grow	Yes	Yes	Yes	Yes
	Online Shrink	No	No	Yes	No
	Offline Grow	Yes	Yes	No	No
	Offline Shrink	Yes	No	No	No
RAID Solution Provided		No	No	Yes	Yes
Volume Manager		LVM/MD	LVM/MD	LVM/MD or BtrFS	ZFS
RBD support		Yes	Yes	Yes	Yes

Figure 2: Flexibility and ease of use features for Linux file systems [7]

ZFS is considered as a complete storage stack solution while Ext4 and XFS is only one part of a storage stack solution. Ext4 permits online and offline grow its file system but still have to unmount the file system first before shrinking its file system size. XFS only permits online to grow but no shrink feature. BtrFS provides the best write and rewrite throughput compared to other file systems like Ext4, XFS, and ZFS. Ext4 performs best as compare to others system in operations like creates, deletes, appends, reads writes and attribute operations on a directory tree. Hence Ext4 would be a good candidate for the file server. BtrFS and ZFS have their volume manager software incorporated in their solutions thus made the deployment of volumes is simple and quick. Figure 2 describes the summary of flexibility and ease of use features for Linux file system.

4.2.1 How Linux Manages Storage Devices - Neha

[6]In Linux, almost everything is represented by a file. Hardware like storage drives, which are represented on the system as files in the /dev directory. Generally, files stand for storage

devices start with sd or hd followed by a letter. For example, the first drive on a server is similar to /dev/sda. Partitions on these drives also have files under /dev, which are represented by appending the partition number to the end of the drive name. For instance, the first partition on the drive from the previous example would be /dev/sda1. The Linux kernel decides which device gets which name on each boot. This could be confusing where your devices change device nodes. To resolve this issue, the /dev/disk directory contains sub-directories corresponding with different, more persistent ways to identify disks and partitions on the system. These contain symbolic links that are generated at boot back to the correct /dev/[sh]da* files. The links are named according to the directory's identifying trait (for instance, by partition label in for the /dev/disk/bypartlabel directory). These links will always point to the correct devices, so they can be used as static identifiers for storage spaces.

4.2.2 Mounting Block Devices - Neha

Mounting is simply the process of attaching a formatted partition or drive to a directory within the Linux file system. Drives are typically mounted on dedicated empty directories. Many different mounting options can be set to change the behavior of the mounted device. For instance, the drive can be mounted in read-only mode to ensure that its contents won't be altered. The file system Hierarchy Standard suggests using /mnt or a sub-directory under it for temporarily mounted file systems.

4.2.3 Making Mounts Permanent with /etc/fstab - Neha

Linux systems refer a file called /etc/fstab (file system table) to decide which file systems to mount during the boot process. File systems that do not have an entry in this file will not be automatically mounted.

4.3 File System on Mobile OS - Liam

Since mobile phones are constrained by size, mechanical drives are simply too big, too expensive in power consumption to be considered. In contrast, flask memory uses less energy, provides fast read access time, and has better shock resistance to withstand the daily abuses from users. In the mobile phone space, there are two major types of flash memory are used: NOR and NAND.
NOR flash memories have been increasingly used in embedded devices for a very long time due to their reliable storage. It supports execute in place (XIP) method for executing code directly from flash memory without copying the code to RAM. [8] To use this method, usually, the memory must provide a similar interface to the CPU as regular memory and provides sufficient fast read operations with a random access pattern. This precisely is the property of NOR flash. [9] They are low density offers fast reads but slow writes. On the other hand, its sibling NAND flash is a low cost, high density, and offers fast writes and slow reads. Increasingly, mobile devices use both NAND flash for storage and NOR for code and execution. An important limitation on NAND memory is block erasure and memory wear. Block erasure means that when erasing any memory, the whole block must be erased. To

overcome this limitation, memory segments to be removed are first marked as dirty. When the whole block is dirty, then it can be erased. Also, memory wear is another big issue. In general, flash memories have a finite program and erase (P/E) cycle limits. [10] To combat this, wear leveling reduction techniques are used to evenly distribute the burden of repeated P/E cycles over a large set of blocks. These techniques are incorporated into an open-source file system called Yet Another Flash File System (YAFFS) which used in Android OS as well.

4.3.1 YAFFS - Liam

[2] YAFFS file system was developed by Toby Churchill Ltd as a reliable system with fast boot time for flash memory devices. It was originally an attempt to modify Journaling Flash File System (JFFS) which was used for NOR to add NAND support. However, it led to slow boot time and high RAM consumption. This was due to substantial differences between NOR and NAND. As a result, they designed a different flash system especially for NAND only. Upon completion, the file system works great on NAND flash but it performs better than other file systems in NOR flash as well. YAFFS has a log-structure file system, which is robust against power failure. It does so by using some RAM to maintain its data structure. It also has a highly optimized and predictable garbage collection strategies that make it performant and deterministic under hard writing. When free space becomes low, a block with some dirty pages and some good pages will be chosen. YAFFS will move the good pages to another block, then the whole block is erased. In addition, it has some desirable features such as low power requirement, flexible in the type of flash it can work on, portable, and robust.

5 Power Management

5.1 Power Management in Linux - Neha

[11] Two Power Management Standards are advanced power management (APM) and the advanced configuration and power interface (ACPI). APM is a proposed by Microsoft and Intel for system power management. APM consists of one or more layers of software to support power management. APM standardizes the information flow across those layers. In the APM model, BIOS plays important role. ACPI is the latest of the two technologies.ACPI is a specification by Toshiba, Intel, and Microsoft for defining power management standards. ACPI allows for more intelligent power management, as it is managed more by the OS than by the BIOS.Important gals of power management software is to keep all devices in their low power states as much as possible. It manages state transitions in association with device drivers and applications. Device drivers are in charge for saving device states before putting them into their low power states and also for restoring the device state when the system becomes active.what device drivers need to do to participate in power management?

- pm_dev structure: The PM subsystem in the Linux kernel keeps some information in a pm_dev structure about every registered driver. Maintaining this information permits it to notify all registered drivers about PM events.

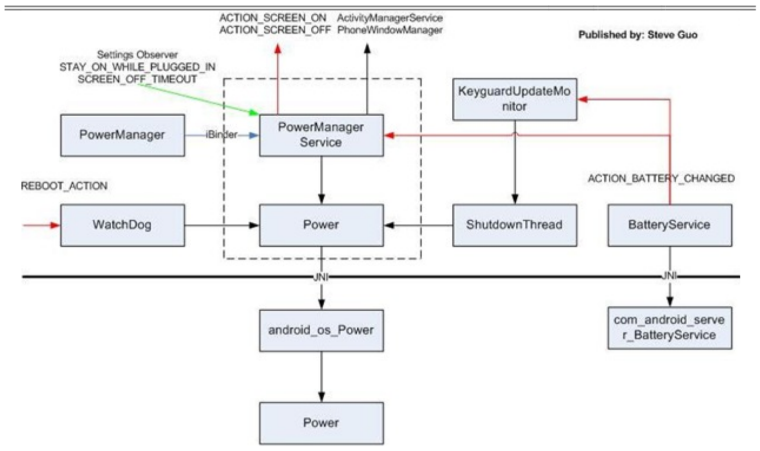

Figure 3: Android Power Management [2]

- pm_register: device drivers first have to register themselves with the PM subsystem before participating in power management.

The linux/pm.h file defines the various types and IDs that can be used by drivers. If successful, pm_register returns a pointer to a structure of type pm_dev. A driver's callback function is invoked by the PM subsystem whenever there is a PM event.Each device driver is supposed to do some processing according to the PM event type. The callback function should return an integer value. Returning a value of zero signifies that the driver agrees to the PM event. A nonzero value signifies that the driver does not agree to the PM event. This may cause the state transition in progress to be aborted.All the driver's callback functions are invoked in a predefined order. This is on a last-come-first-served basis, which can be a problem if two devices depend on each other. A driver stops participating in power management by calling pm_unregister.To unregister, it has to supply the pointer to the same function it used while registering. Once a driver has unregistered itself, the PM subsystem stops involving it in further PM events.

5.2 Power Manage in Mobile OS - Liam

In contrast to the standard Linux system, Android does not use APM, nor ACPI for power management. Instead, it use PowerManager instead, from figure 3. The idea is that Android requires applications and services to request CPU resources with wake-locks through the Android application framework. If there is no active wake lock, Android will shut down the CPU to conserve battery life [12]. This module provides low level drivers to control the peripherals which include screen and keyboard. Also, PowerManager monitors the battery life and status of the device. This allows the manager to take control of the charging of the battery and limit power consumption when the battery level has reached a critical threshold. There are 4 different flag values for wake locks as seen in table 4.

Table 1 Wake Lock Settings [26]

Flag Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On*	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

Figure 4: Wake locks in Android OS [2]

Table 1. Permission character symbols

Symbol	Description
r	Indicates that a given category of user can **read** a file.
w	Indicates that a given category of user can **write** to a file.
x	Indicates that a given category of user can **execute** the file.
-	A fourth symbol indicates that no access is permitted.

Table 2. Permission categories

Category	Description
Owner	The owner of the file or application.
Group	The group that owns the file or application.
Everyone	All users with access to the system.

Figure 5: Permissions for files, directories, and applications [13]

6 Security

6.1 Security in Linux - Neha

[13] For the basic security features, Linux has password authentication, file system discretionary access control, and security auditing. These three fundamental features are necessary to achieve a security evaluation at the C2 level. Most commercial server-level operating systems, have been certified to this C2 level. By expanding the basic standard security features we have: User and group separation, File system security, Audit trails, PAM authentication.

6.1.1 User and group separation - Neha

[14] Linux was designed to permit more than one user to have access to the system at the same time. Permissions are nothing but the “rights” to act on a file or directory. The basic rights are read, write, and execute.User accounts are used to verify the identity of the person using a computer system. Groups are logical constructs that can be used to group user accounts for a particular purpose. The ability for a user to access a machine is decided by whether or not that user’s account exists. Access to an application or file is granted based on the permission settings for the file. This helps to ensure the integrity of sensitive information and key resources against accidental or purposeful damage by users. To view the permissions on a file or directory, issue the command ”ls -l”

6.1.2 File System Security - Neha

A very true statement of a Linux system, everything is a file; if something is not a file, it is a process. There are some exceptions as listed in figure 6. On the Linux system, every file is owned by a user and a group user. For each category of users, read, write, and execute permissions can be granted or denied. After execution of command ”ls -l”. The permissions

Symbol	Meaning
-	Regular file
d	Directory
l	Link
c	Special file
s	Socket
p	Named pipe
b	Block device

Figure 6: Characters determining the file type [13]

Code	Meaning
0 or -	The access right that is supposed to be on this place is not granted.
4 or r	read access is granted to the user category defined in this place
2 or w	write permission is granted to the user category defined in this place
1 or x	execute permission is granted to the user category defined in this place

Figure 7: Access mode codes [13]

are always in the same order: read, write, execute for the user, the group, and the others. Both access rights or modes and user groups have a code shown in figure 7 and 8.
We can use the chmod command to modify the file permission, changing the access mode of a file. The chmod command can be used with alphanumeric or numeric options, whichever is suitable for you. The + and - operators are used to grant or deny a given right to a given group. When using chmod with numeric arguments, the values for each granted access right have to be counted together per group. Thus we have a 3-digit number, which is the symbolic value for the settings chmod has to make. The figure 9 lists the most common combinations.

6.1.3 Audit Trails - Neha

[15] Linux kernel 2.6 comes with auditd daemon. It's responsible for writing audit records to the disk. while booting, the rules in /etc/audit.rules are read by auditd daemon. we can open /etc/audit.rules file and make changes such as setup audit file log location and other option. In order to use audit facility, we need to use utilities mentioned in figure 10

6.1.4 Pluggable Authentication Modules authentication (PAM) - Neha

[16] Linux-PAM provides a flexible mechanism for authenticating users. PAM consists of a group of libraries that manage the authentication tasks of applications on the system. PAM is a more versatile user authentication mechanism. Programs supporting PAM must dynamically link themselves to the modules in charge of authentication. The administrator is in charge of the configuration and the attachment order of modules. All applications using PAM must have a configuration file in /etc/pam.d. Each file is composed of four columns mentioned in figure 11

Code	Meaning
u	user permissions
g	group permissions
o	permissions for others

Figure 8: User Group Code [13]

Command	Meaning
chmod 400 file	To protect a file against accidental overwriting.
chmod 500 directory	To protect you from accidentally removing, renaming or moving files from this directory.
chmod 600 file	A private file only changeable by the user who entered this command.
chmod 644 file	A publicly readable file that can only be changed by the issuing user.
chmod 660 file	Users belonging to your group can change this file; others don't have any access to it at all.
chmod 700 file	Protects a file against any access from other users, while the issuing user still has full access.
chmod 755 directory	For files that should be readable and executable by others, but only changeable by the issuing user.
chmod 775 file	Standard file sharing mode for a group.
chmod 777 file	Everybody can do everything to this file.

Figure 9: File protection with chmod [13]

Utility	Description
auditctl	A command to assist controlling the kernel's audit system. You can get status, and add or delete rules into kernel audit system
ausearch	A command that can query the audit daemon logs based for events based on different search criteria.
aureport	A tool that produces summary reports of the audit system logs.

Figure 10: Audit Utility [13]

Column	Description
Module type	▪ auth: user authentication ▪ account: user restriction (e.g.: hour restriction) ▪ session: tasks to perform at login and logout e.g.: mounting directories ▪ password: update of the user authentication token
success control	▪ required: a least one of the required modules ▪ requisite: all the requisite modules ▪ sufficient: only one sufficient module ▪ optional: a least one of the required modules is necessary if no other has succeeded
path to the module	Usually /lib/security.
optional arguments	-

Figure 11: PAM's pam.d content [13]

6.2 Mobile OS Security - Liam

[4] The underlying security system of mobile OS is dependent on the users' capabilities to access the system. In this case, each application runs with its unique user id and in its process since DVM can run multiple instances. Since all DMV application runs in a sandbox, it prohibits default access to other processes such as GPS, hardware features unless allowed. Since Android uses a virtual machine system, it's easy for DVM to enforce these constrain to enforce certain application behavior. Some other OS such as IOS or webOS with no virtual machine, their sandboxing system is based on the kernel, file system or process level to enforce these constraints.

[17] In Android, several security mechanisms are incorporated into the Android Framework. Every Android package (.apk) file installed on a device has a unique user ID through Linux (POSIX) ID. Because of this, two different packages with the same ID can't run in the same process. Enabling sharedUserID feature enables two applications to share the same permission sets and run in the same process. However, both applications must be digitally signed by the same author. The signed apk is valid as long as its certificate is. valid and the enclosed public key successfully verifies the signature.

With the recent push to protect user's privacy, Android permission mechanisms enforce many restrictions on privacy operations. Currently, Android has roughly 100 built0in permissions that control operations. Users must explicitly allow these permissions to enable it. Permissions are grouped by protection levels ranging from normal, dangerous, signature, and signature-or-system.

In a recent survey of Android Framework Security Assessment, it's reported that a normal

Android device is well protected against attackers from replacing the core components or the kernel. The only way to alter the operating system components is to exploit vulnerabilities in the kernel modules or core libraries to enable root access. [17] One weak point of Android OS is WebKit, an open source Web engine. Some recent attacks on it include buffer overflow in an outdated native library and an explicit cross-site scripting (XSS) vulnerability. These attacks can use the abilities and privileges of the browser applications to gain root access. There are some concerns about Bluetooth vulnerabilities as well but the survey believes it's less likely to occur.
[4] Another weak point is Java Native Interface (JNI). When applications need native code outside DVM, JNI can be used to includes features that are not available through DVM. The code from the native library is loaded into the address space of the application virtual machine instances. This can leads to a major security hole that can be abused. Therefore, in the recent revision of Native Development Kit, code used by native applications should only make use of limited libraries.

7 Conclusion - Neha, Liam

As traditional operating systems run on convectional desktops/ laptops/servers it focuses on delivering speed with power full CPU. Linux operating system supports more than twenty hardware architectures. Linux kernel support loading and removing modules based on user's requirements which provide flexibility to the operating system. Linux support multiple file system like Ext4, XFS, BtrFS, and ZFS.[7]BtrFS has shown significant I/O performances compared to other file systems for write and re-write throughput. In terms of flexibility and ease of use, BtrFS and ZFS would be a good choice. So the user can select a file system according to his requirement. Linux system is managed by either Advanced Power Management (APM) or Advanced Configuration and Power Interface (ACPI). Although APM has some drawbacks, its simplicity allows it to be implemented in almost any device. On the other hand, ACPI provides richer control over power management at the cost of complexity. Linux provided a highly secure environment. It has multiple features to provide a secure environment for users. Hence we can answer "why is Linux the leading operating system among other traditional operating systems?"
Mobile devices are a type of embedded device that operates within a restrictive environment. They are simply underpowered to use a traditional OS, but they do share common operations as a traditional computer. Therefore, Android borrows kernel design from Linux with its implementations on top. Some of the implementations are about optimizations to reduce the footprint of the system and maximize the types of workloads it handles. Some are for the differences in hardware the OS is running on such as flash memory, battery power, and touch screen. Some of the modifications are there to control and protect the only user of the systems and prevent the exposure of user's privacy. With the recent improvements of the processor and price reduction of flash memory, mobile phones are closing the gap in computing power to the traditional desktop. There is a potential in the future where it might be indistinguishable between a mobile OS and a traditional OS.

References

[1] Richard Fox. Linux with operating system concepts. URL http://docshare01.docshare.tips/files/31762/317621784.pdf. Online.

[2] Yu-Hsuan Chan Frank Maker. A survey on android vs. linux. *Department of Computer Science, University of California, Davis*.

[3] Mathieu Devos. Bionic vs glibc report master thesis. *IRATI Investigating Rina*, 2013-2014.

[4] Dominique A. Heger. Mobile devices - an introduction to the android operating environment design, architecture, and performance implications.

[5] Somya Lal Abhyudai Shanker. Android porting concepts.

[6] An introduction to storage terminology and concepts in linux, . URL https://www.digitalocean.com/community/tutorials/an-introduction-to-storage-terminology-and-concepts-in-linux.

[7] M. B. Ab Karim, J. Luke, M. Wong, P. Sian, and O. Hong. 2016 ieee conference on open systems (icos). pages 18–23, 2016.

[8] Sriram Balasubramanian Sreenath Panganamala, Nishith Sharma. Achieving high performance non-volatile memory access through "execute-in-place" feature, 2020. URL https://www.design-reuse.com/articles/41861/execute-in-place-xip-nor-flash-spi-protocol.html. Online.

[9] Steve Lowry. 2 types of flash memory — nor flash and nand flash, 2020. URL https://www.gearbest.com/blog/how-to/2-types-of-flash-memory-nor-flash-and-nand-flash-2823. Online.

[10] Wear leveling in nand flash memory. *Macronix International LTD*, 2014.

[11] Linux power management, . URL https://www.linuxjournal.com/article/6699.

[12] Power management, 2020. URL https://wladimir-tm4pda.github.io/porting/power_management.html. Online.

[13] Linux security, . URL https://www.tenouk.com/linuxunixsecurityfeatures.htmls.

[14] Linux users and groups, . URL https://www.linode.com/docs/tools-reference/linux-users-and-groups/.

[15] Audit trails, . URL https://sematext.com/blog/auditd-logs-auditbeat-elasticsearch-logsene/.

[16] Pam, . URL https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/managing_smart_cards/pluggable_authentication_modules.

[17] Uri Kanonov Yuval Elovici Shlomi Dolev Chanan Glezer Asaf Shabtai, Yuval Fledel. Google android: A comprehensive security assessment. *Mobile Device Security*, 2010.