



operating system
Spring 2017

prof. dr. asan alik

Student Name : *Asan Alkan*

Student ID : 163110450

Email : a.ramadan1976@yahoo.com



Android (Marshmallow or Nougat) Process and Thread Management

Introduction

There is the latest update of Android is called as Android 6.0 “Marshmallow” to the operating system. Its third & final preview released on August 7, 2015. Google has rolled out with an update is that Android M is now called as Android 6.0 Marshmallow. The primary focus of this new emergence is to enhance the overall user experience as in it will bring a few features, including “redesigned permission model” in that at the time of installation, the applications will no longer automatically granted all of their specified permissions.

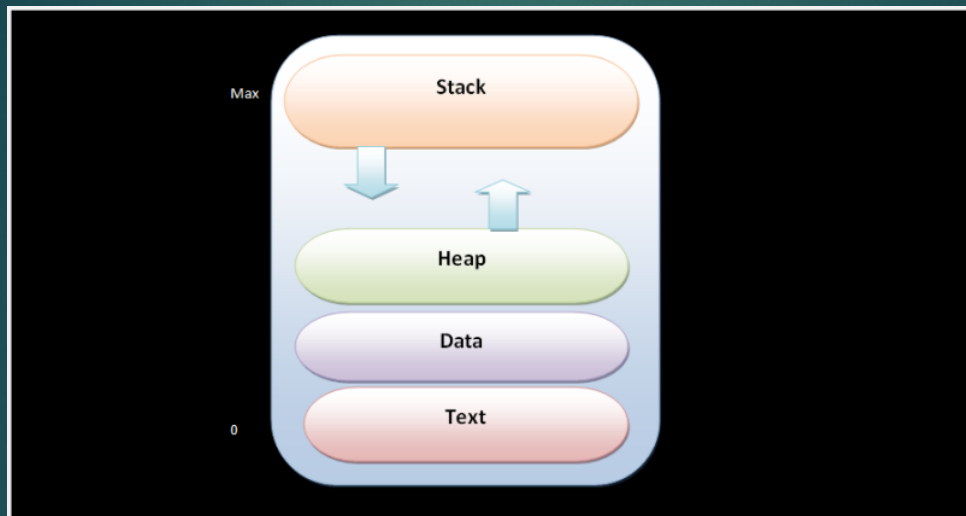
Processes

Process is a program in execution. A process has a life cycle.

Earlier computer systems allowed only one program to be executed at a time. This program ruled the system. It had access to all resources. Nowadays, our systems are time sharing. Multiple programs can be loaded together. They execute concurrently. Here comes the role of process. A process is a program in execution. To perform better, operating systems have a collection of processes.

- ▶ System Processes execute system code.
- ▶ User Processes executes user code.

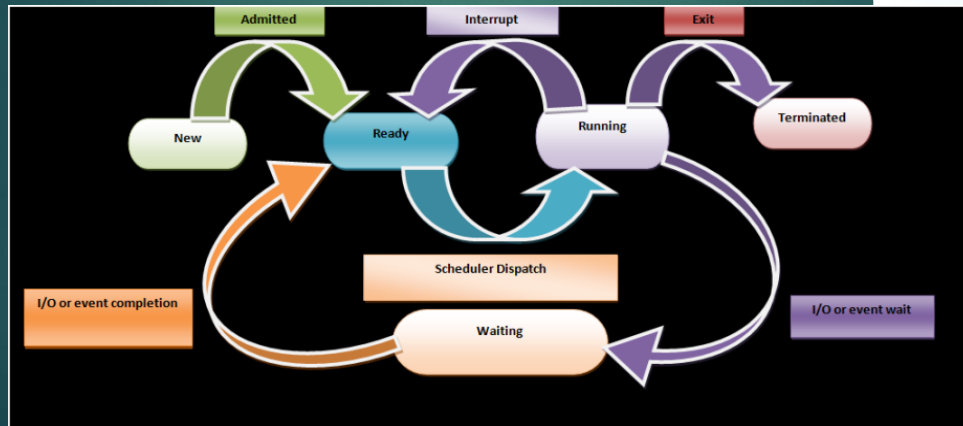
These processes can run concurrently, which is achieved by switching the CPU between processes. Structure of a process is as shown below:



Structure of a process

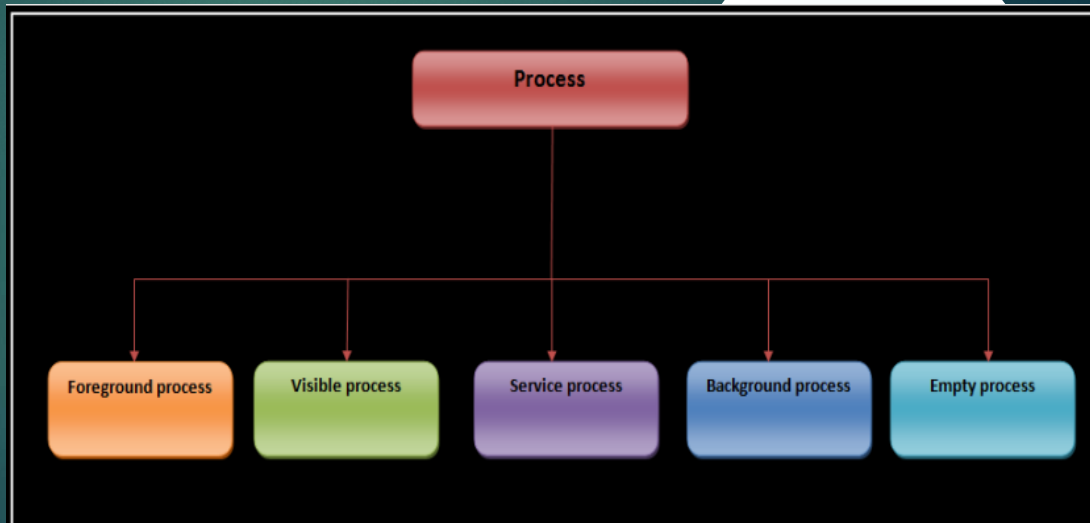
Process States

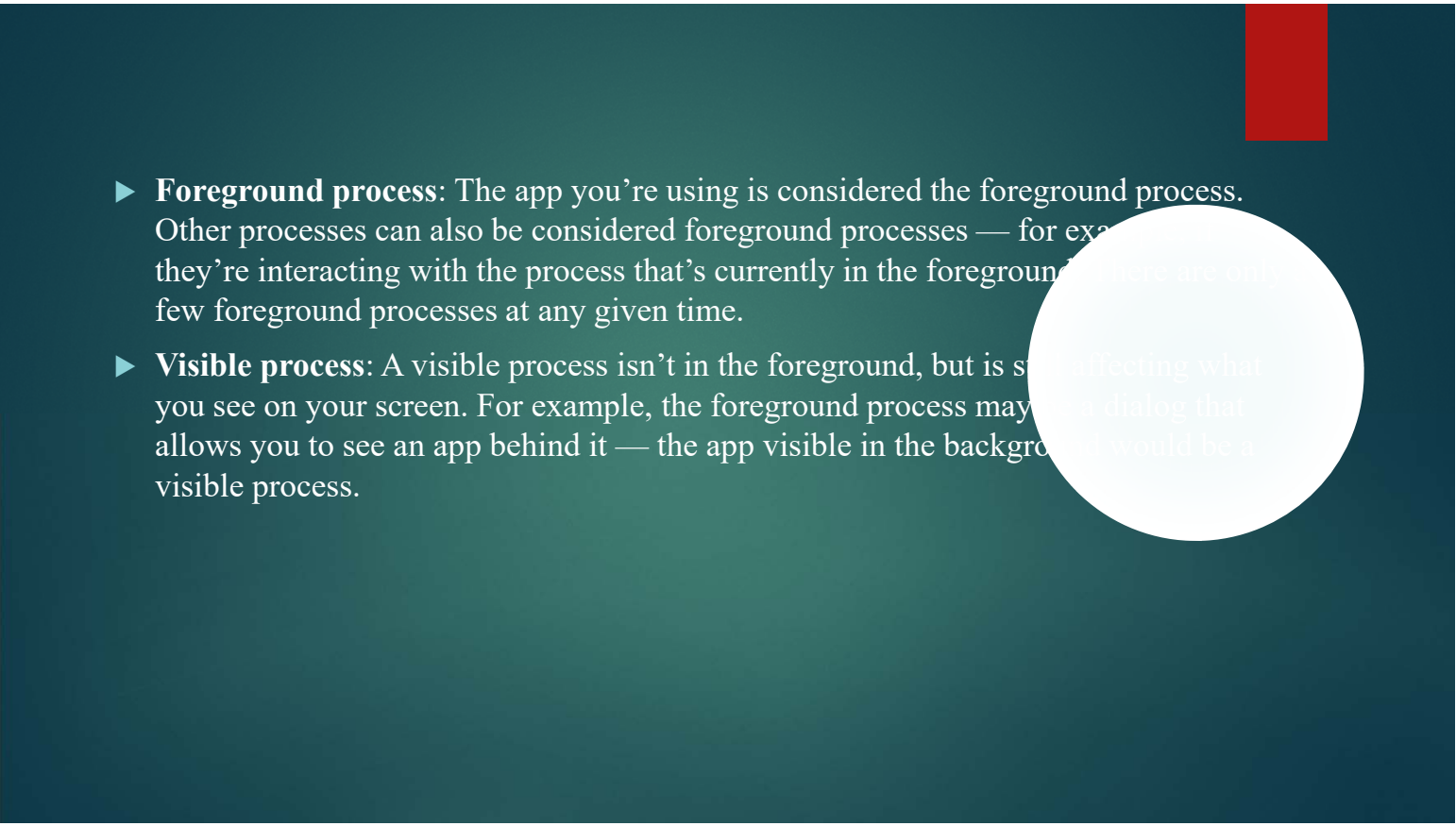
- ▶ **New:** The process is created.
- ▶ **Running:** Instructions are executed at this stage.
- ▶ **Waiting:** Process is waiting for some event to occur.
- ▶ **Ready:** Process is waiting to be assigned to a processor.
- ▶ **Terminated:** Execution is finished.

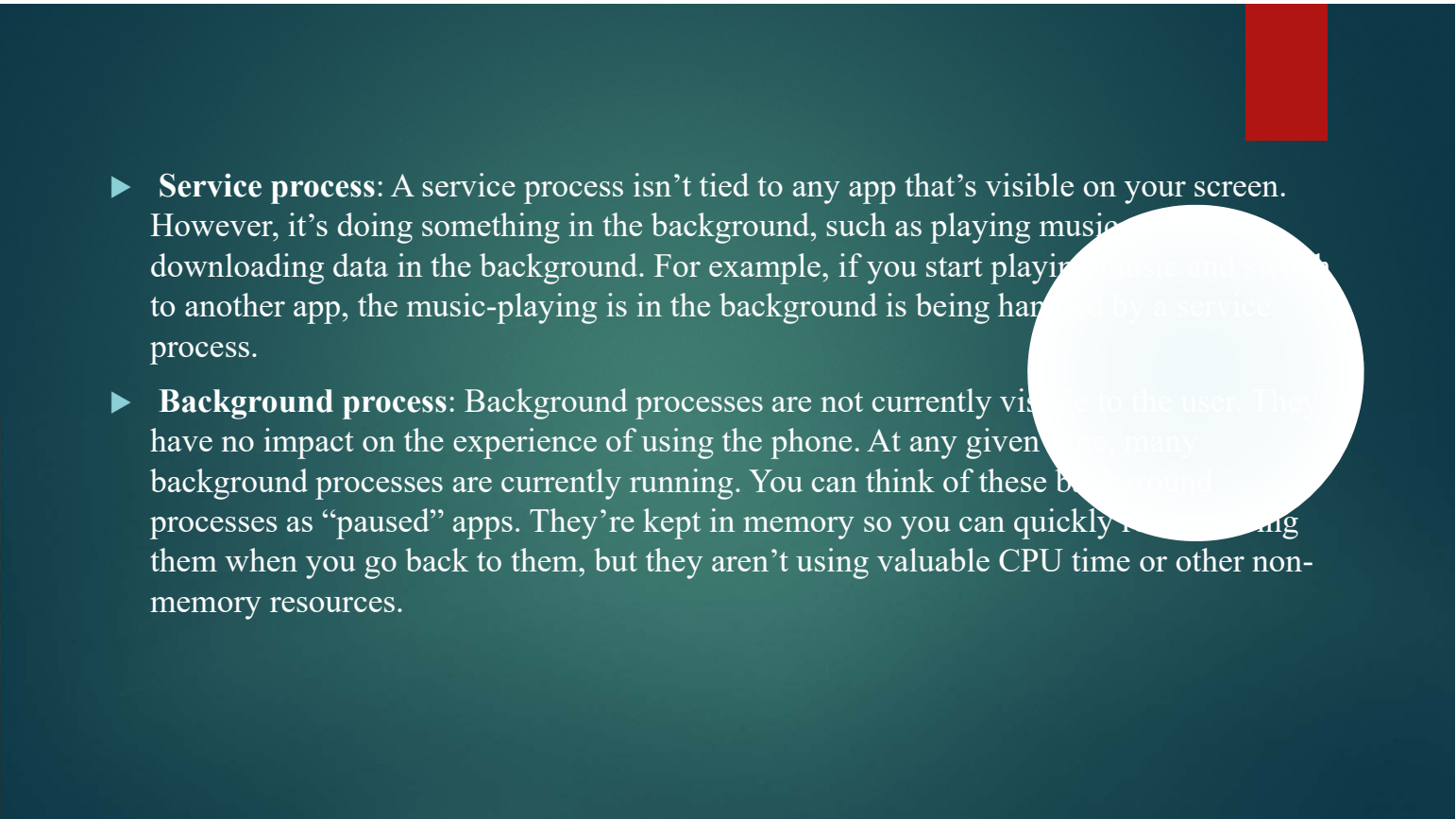




▶ **Android** system maintains processes as long as possible but sometimes they have to kill processes to recover resources. This killing procedure is decided by the importance hierarchy. Lowest priority gets out of the system first and processes with higher priority will be eliminated last. We already know about these processes. Their importance hierarchy is as shown below:

- ▶ Foreground Process
- ▶ Visible Process
- ▶ Service Process
- ▶ Background Process
- ▶ Empty Process.



- 
- ▶ **Foreground process:** The app you're using is considered the foreground process. Other processes can also be considered foreground processes — for example, if they're interacting with the process that's currently in the foreground. There are only a few foreground processes at any given time.
 - ▶ **Visible process:** A visible process isn't in the foreground, but is still affecting what you see on your screen. For example, the foreground process may show a dialog that allows you to see an app behind it — the app visible in the background would be a visible process.

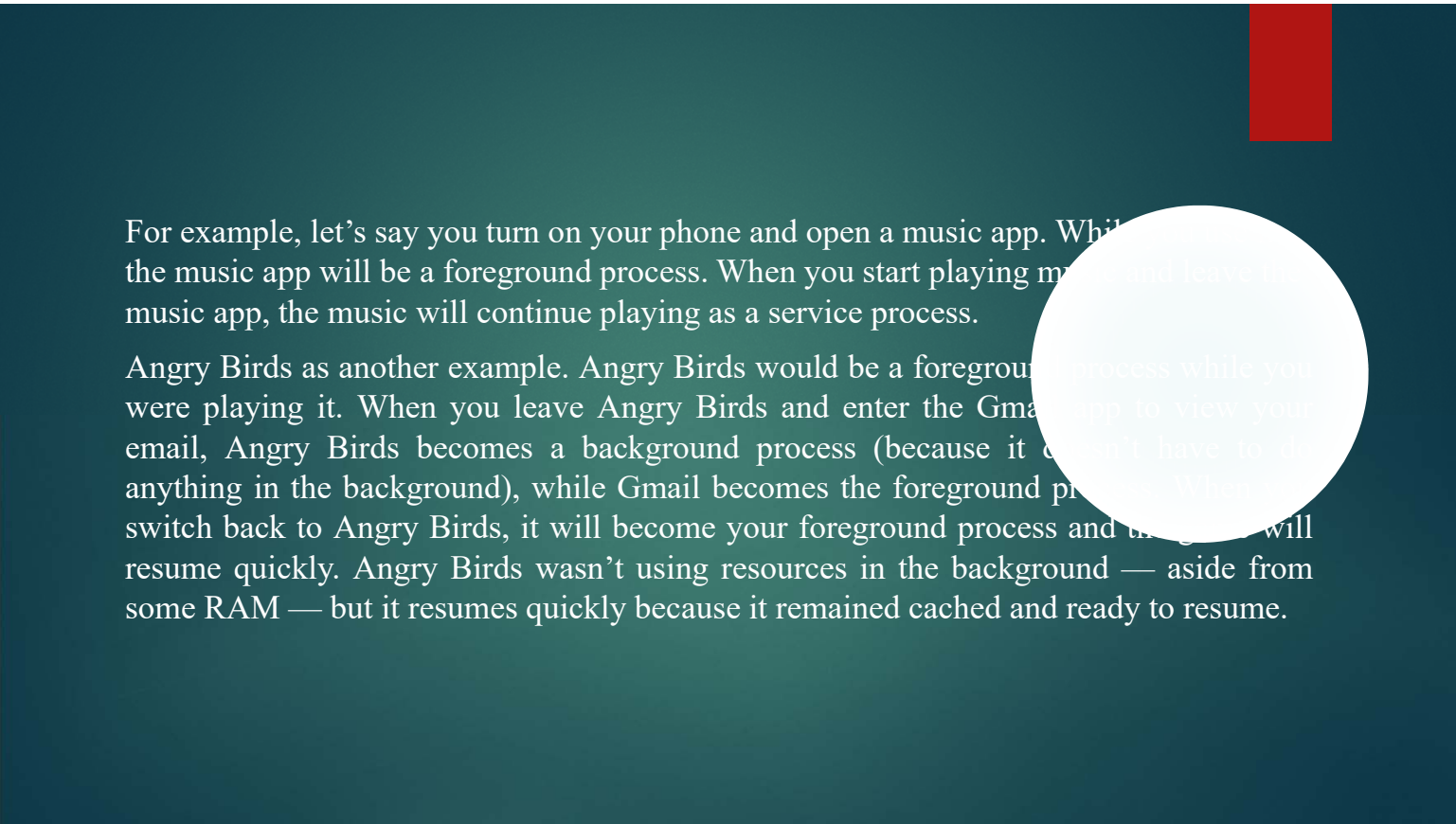
- 
- ▶ **Service process:** A service process isn't tied to any app that's visible on your screen. However, it's doing something in the background, such as playing music or downloading data in the background. For example, if you start playing music and switch to another app, the music-playing in the background is being handled by a service process.
 - ▶ **Background process:** Background processes are not currently visible to the user. They have no impact on the experience of using the phone. At any given time, many background processes are currently running. You can think of these background processes as “paused” apps. They're kept in memory so you can quickly resume using them when you go back to them, but they aren't using valuable CPU time or other non-memory resources.

- 
- 
- ▶ **Empty process:** An empty process doesn't contain any app data anymore. It can be kept around for caching purposes to speed up app launches later, or the system may kill it as necessary.



Android Automatically Manages Processes

- ▶ When Android needs more system resources, it will start killing the least important processes first. Android will start to kill empty and background processes to free up memory if you're running low. If you need more memory — for example, if you're playing a particularly demanding game on a device without much RAM, Android will then start to kill service processes, so your streaming music and file downloads may stop.
- ▶ Android provides apps with so much flexibility that they have room to misbehave. For example, a poorly coded app could start a service process that remains running in the background all the time, using up all your CPU time and dramatically decreasing your battery life.

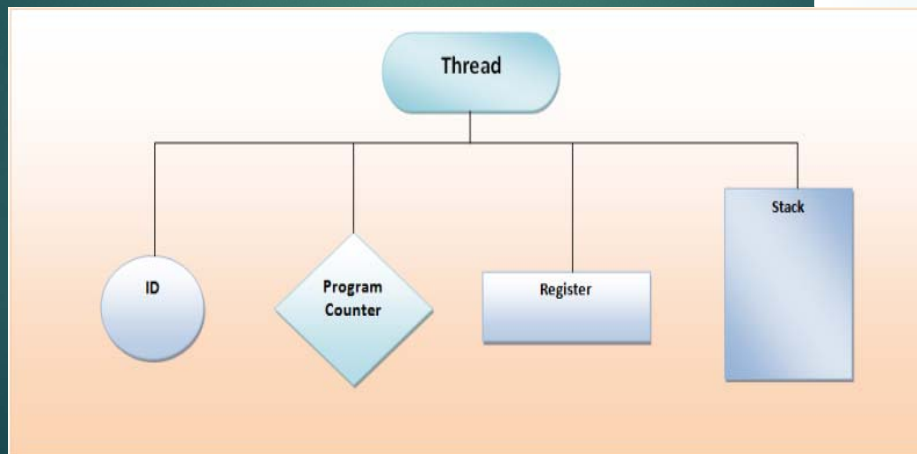


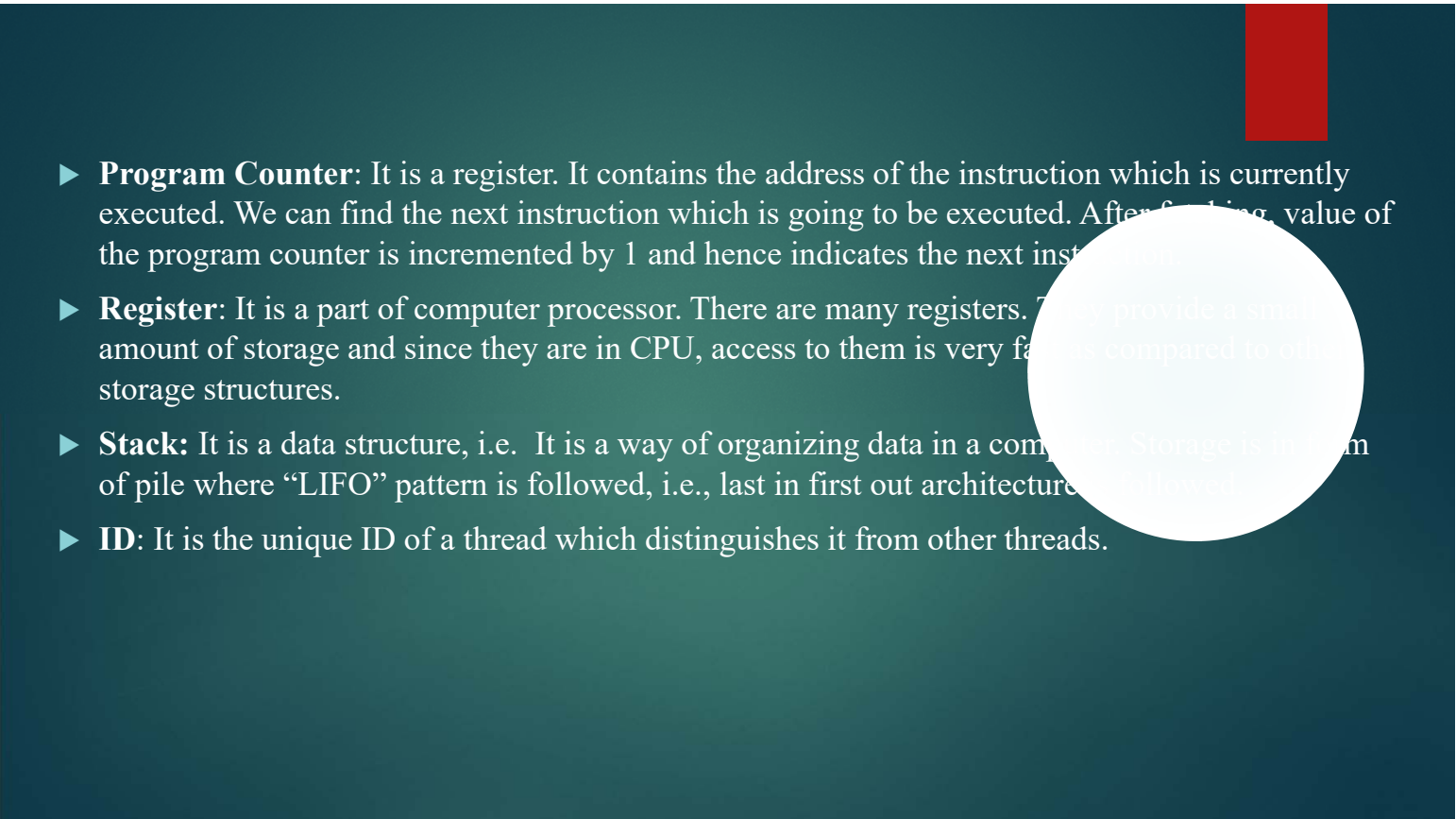
For example, let's say you turn on your phone and open a music app. While you use the music app, the music app will be a foreground process. When you start playing music and leave the music app, the music will continue playing as a service process.

Angry Birds as another example. Angry Birds would be a foreground process while you were playing it. When you leave Angry Birds and enter the Gmail app to view your email, Angry Birds becomes a background process (because it doesn't have to do anything in the background), while Gmail becomes the foreground process. When you switch back to Angry Birds, it will become your foreground process and Angry Birds will resume quickly. Angry Birds wasn't using resources in the background — aside from some RAM — but it resumes quickly because it remained cached and ready to resume.

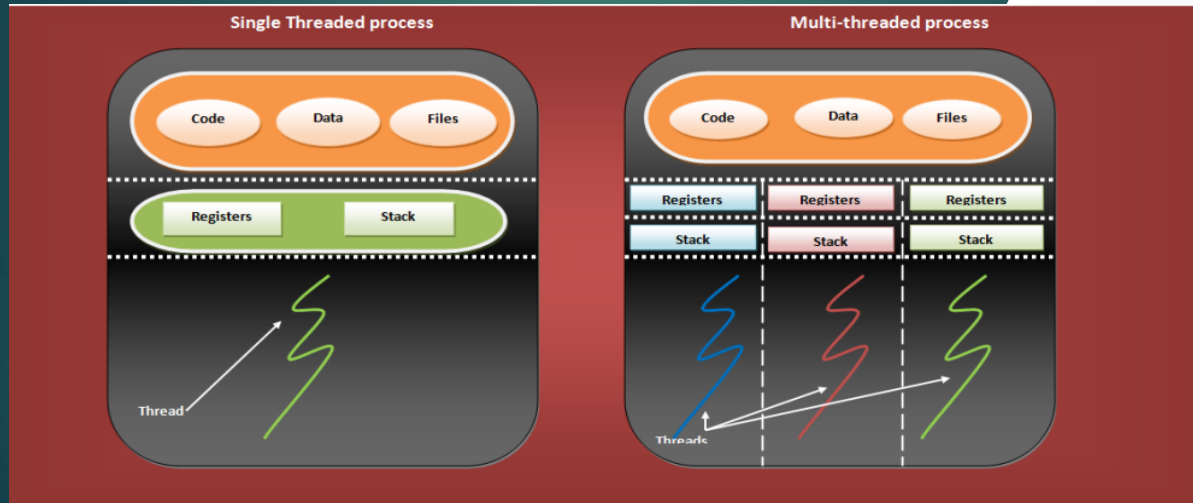
Android Threads

A **thread** is the smallest unit of processing which can be managed by the CPU scheduler. Generally, threads are contained in a process. Thread is a friendly guy who comes to point of sharing resources with other threads i.e., threads belonging to same process can share their resources even the operating system resources. A thread has its own ID, a program counter, a register and a stack



- 
- ▶ **Program Counter:** It is a register. It contains the address of the instruction which is currently executed. We can find the next instruction which is going to be executed. After finishing, value of the program counter is incremented by 1 and hence indicates the next instruction.
 - ▶ **Register:** It is a part of computer processor. There are many registers. They provide a small amount of storage and since they are in CPU, access to them is very fast as compared to other storage structures.
 - ▶ **Stack:** It is a data structure, i.e. It is a way of organizing data in a computer. Storage is in form of pile where “LIFO” pattern is followed, i.e., last in first out architecture is followed.
 - ▶ **ID:** It is the unique ID of a thread which distinguishes it from other threads.

- ▶ There can be multiple threads in a process. Threads facilitate concurrency or multi-tasking in operating systems. For example, Word may have one thread to display graphics and other for checking spelling or grammar mistakes, etc. A process can contain single thread or multiple threads. Let us see the difference between two of them.



Advantages of Threads

- ▶ **Receptiveness:** If an interactive application like a web browser can be multi-threaded then, even if a part of the program is blocked or busy performing any long running task, the application responds to new requests quickly and hence increases the receptiveness of the application.
- ▶ **Providence:** Threads share resources and hence they are very economical. Allocating memory and resources is just like investing money and thus they reduce the wastage of resources. It is easy to manage threads as compared to processes.
- ▶ **Sharing:** Threads share resources by default. They share the code and data and thus applications can share several threads of activity within the same address space.
- ▶ **Architecture:** Multi-threaded programming increases the usability of multi-processor architecture.
- ▶ **Distributed Applications:** They are very useful in distributed applications as well. For e.g., the server will be multi-threaded one for each client. In fact clients too are multi-threaded, one for managing the connection with server and other for communicating with the server.

Thank you

