# OS CONCEPTS

Operating System Concepts

Threads

Definition: A thread is the smallest unit of a process that can be scheduled and executed by the operating system. Threads share the same process resources but operate independently.

Types of Threads:

1. User Threads: Managed by a user-level library and not directly by the kernel. These are lightweight but lack kernel-level optimizations.

2. Kernel Threads: Managed directly by the operating system kernel, providing better performance and more control over thread management.

3. Hybrid Threads: Combine the benefits of user and kernel threads. Both user and kernel-level thread management is utilized.

How Threads Work in Android:

- Main Thread: Also known as the UI thread, it's responsible for handling all UI operations and interactions. Long-running tasks on the main thread can lead to application not responding (ANR) errors.

- Worker Threads: Used for performing background operations like network requests, database operations, etc. These threads help keep the UI thread responsive.

Thread Management in Android:

- AsyncTask: Simplified API for short background tasks that need to interact with the UI.

- Handler and Looper: Provides a way to send and process `Message` and `Runnable` objects associated with a thread's `MessageQueue`.

- Executors: A higher-level API for managing thread pools.

Scheduling Algorithms

Definition: Scheduling algorithms determine the order in which processes or threads are executed by the CPU.

Common Scheduling Algorithms:

1. First-Come, First-Served (FCFS):

- Jobs are executed in the order they arrive.

- Simple but can lead to long waiting times (convoy effect).

2. Shortest Job Next (SJN) or Shortest Job First (SJF):

- Jobs with the smallest execution time are executed first.

- Optimal for minimizing waiting time but requires knowledge of execution times in advance.

3. Priority Scheduling:

- Jobs are assigned priorities, and the highest priority job is executed first.

- Can lead to starvation if lower priority jobs are perpetually postponed.

4. Round Robin (RR):

- Each job is given a fixed time slice or quantum.

- Jobs are executed in a cyclic order, ensuring fair sharing of CPU time.

- Suitable for time-sharing systems.

5. Multilevel Queue Scheduling:

- Multiple queues for different priority levels.

- Each queue can have its own scheduling algorithm.

6. Multilevel Feedback Queue Scheduling:

- Similar to multilevel queue scheduling but allows jobs to move between queues based on their execution behavior and requirements.

Threads and Scheduling in Android

In Android, proper management of threads is crucial for creating responsive and efficient applications.

Example of AsyncTask:

```java

private class DownloadTask extends AsyncTask<URL, Integer, Long> {

protected Long doInBackground(URL... urls) {

int count = urls.length;

long totalSize = 0;

for (int i = 0; i < count; i++) {

totalSize += Downloader.downloadFile(urls[i]);

publishProgress((int) ((i / (float) count) 100));

}

return totalSize;

}

protected void onProgressUpdate(Integer... progress) {

setProgressPercent(progress[0]);

}

protected void onPostExecute(Long result) {

showDialog("Downloaded " + result + " bytes");

}

}

```

Example of Handler and Looper:

```java

Handler handler = new Handler(Looper.getMainLooper()) {

@Override

public void handleMessage(Message msg) {

// Process the message

}

};

// To post a message to be processed by the handler

handler.post(new Runnable() {

@Override

public void run() {

// Code to run on the main thread

}

});

```

Example of Executor Service:

```java

ExecutorService executor = Executors.newFixedThreadPool(4);

executor.execute(new Runnable() {

@Override

public void run() {

// Background task

}

});

executor.shutdown();

```

Summary

- Threads are the smallest units of execution within a process. They can be user, kernel, or hybrid threads.

- In Android, proper thread management is essential for responsive apps, using tools like `AsyncTask`, `Handler`, and `ExecutorService`.

- Scheduling algorithms determine the execution order of processes/threads, with common types including FCFS, SJF, Priority Scheduling, RR, and Multilevel Queue Scheduling.