转载请注明链接:https://blog.csdn.net/feather_wch/article/details/82556153

HandlerThread

版本：2018/9/9-1(10:10)

问题汇总

> 1. 总结HandlerThread相关所有可能的问题，用于自我检测和查漏补缺。
> 2. 【☆】标记的是补充问题，直接给答案。其余问题答案都在文中。

1. HandlerThread是什么?
2. 【☆】HandlerThread任务是并行处理的?
   > 不是！是串行处理的。如果一个任务执行时间过长会阻塞后续任务。
3. 【☆】HandlerThread不能执行耗时过长的任务?
   > 内部任务是串行处理的。如果一个任务执行时间过长会阻塞后续任务。
4. 【☆】如何通过HandlerThread来执行一个具体任务?
   > 外界需要通过 `Hanlder` 的消息方式来通知 `HandlerThread` 来执行一个具体任务
5. HandlerThread产生的背景?
6. HandlerThread如何退出?
7. HandlerThread的典型使用场景?
8. HandlerThread的基本使用流程?
9. HandlerThread是如何做到一个线程能够一直运行，有新的任务就处理，没有任务就等待?
10. handlerThread.getLooper()必须在handlerThread.start()之后调用?
11. HandlerThread的构造方法
12. HandlerThread的run()做了哪些事情?
13. HandlerThread如何退出？quit/quit或quitSafely
14. HandlerThread的Tid有什么用?
15. 为什么run中有notifyAll，getLooper中有wait?

16. 为什么要设置线程优先级？为什么能防止内存泄漏和解决数据安全问题?
17. 线程优先级设置的两种方法?两种方式的优先级范围?

基本使用(7)

1、HandlerThread是什么?

> 1. 内嵌Looper、Handler、MessageQueue的Thread
> 2. 内部具有队列，任务会串行处理。(如果一个任务执行时间过长，会阻塞后续任务)
> 3. 执行任务：外界需要通过 `Hanlder` 的消息方式来通知 `HandlerThread` 来执行一个具体任务

2、HandlerThread产生的背景?

> 1. 一般线程执行完任务后就会停止，如果有很多任务导致频繁创建和销毁线程，会损耗系统资源。
> 2. 如果子线程需要更新UI，要使用到Handler，此外需要创建Looper等操作，比较繁琐。

3、HandlerThread如何退出?

> 1. `HandlerThread` 的 `run` 方法是 无限循环 执行的，需要通过 `HandlerThread` 的 `quit`或`quitSafely` 方法来终止 线程的执行

4、HandlerThread的典型使用场景?

> 4. HandlerThread典型使用场景是IntentService

5、HandlerThread的基本使用

```
/**============================
 *  创建并开启HandlerThread
 *============================*/
        //创建一个线程,线程名字：handler-thread
        HandlerThread mHandlerThread = new HandlerThread( "handler-thread") ;
        //开启一个线程
        mHandlerThread.start();
/**============================
 *  创建Handler并实现具体任务
 *============================*/
        //在这个线程中创建一个handler对象
        Handler handler = new Handler( mHandlerThread.getLooper() ){
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                //这个方法是运行在 handler-thread 线程中的 ，可以执行耗时操作
                Log.d( "handler " , "消息：  " + msg.what + "  线程：  " + Thread.currentThread()
            }
        };
/**============================
 *  开启任务的执行
 *============================*/
        //在主线程给handler发送消息
        handler.sendEmptyMessage(1) ;
```

6、HandlerThread是如何做到一个线程能够一直运行，有新的任务就处理，没有任务就等待?

> 1. 利用了 `Looper.loop()` 进行无限循环。

7、handlerThread.getLooper()必须在handlerThread.start()之后调用

> 1. 不然因为 `isAlive() = true` ， 会导致直接返回null。无法去构造Handler

```
Handler handler = new Handler(handlerThread.getLooper());
// java.lang.NullPointerException: Attempt to read from field 'android.os.MessageQueue android.
```

源码(7)

1、HandlerThread的构造方法

> 1. 默认的线程优先级为：THREAD_PRIORITY_DEFAULT，标准的app线程优先级
> 2. 可以自定义线程优先级

```
// HandlerThread.java
public HandlerThread(String name) {
    super(name);
    mPriority = Process.THREAD_PRIORITY_DEFAULT;
}
public HandlerThread(String name, int priority) {
    super(name);
    mPriority = priority;
}
```

2、HandlerThread的run()

> 1. 获取Tid，设置优先级，创建Looper
> 2. 回调 onLooperPrepared()，并开启loop()
> 3. getLooper()获取Looper时，会阻塞调用者线程，直到HandlerThread创建了Looper。

```
// HandlerThread.java-线程的run方法
@Override
public void run() {
    // 1、获取到Tid
    mTid = Process.myTid();
    // 2、创建Looper、MessageQueue
    Looper.prepare();
    synchronized (this) {
        // 3、存储，并唤醒从HandlerThread.getLooper()方法而阻塞的线程
        mLooper = Looper.myLooper();
        notifyAll();
    }
    // 4、设置线程优先级
    Process.setThreadPriority(mPriority);
    // 5、回调，进行loop前的准备工作。
    onLooperPrepared();
    // 6、开始loop()
    Looper.loop();
    mTid = -1;
}
// HandlerThread.java-loop()前的准备工作
protected void onLooperPrepared() {
}

// HandlerThread.java-获取到该线程的Looper
public Looper getLooper() {
    // 1、不存活，直接返回null
    if (!isAlive()) {
        return null;
    }
    // 2、在looper创建前，处于阻塞状态。
    synchronized (this) {
        while (isAlive() && mLooper == null) {
            try {
                wait();
            } catch (InterruptedException e) {
            }
        }
    }
    return mLooper;
}

//@hide: 获取Handler或者用调用者线程的looper创建Handler
public Handler getThreadHandler() {
    if (mHandler == null) {
        mHandler = new Handler(getLooper());
    }
    return mHandler;
}
```

3、HandlerThread如何退出？quit/quit或quitSafely

> 1. 调用Looper的退出方法。

```
        // 通过Looper的quit和quitSafely进行退出。
        public boolean quit() {}
        public boolean quitSafely() {}
```

4、HandlerThread的Tid有什么用?

> 1. Thread id，就是线程的ID，表明线程的身份

```
// 1、获取到Tid
mTid = Process.myTid();
public int getThreadId() {
  //2、返回Tid
      return mTid;
}
```

5、为什么run中有notifyAll，getLooper中有wait?

> 1. 避免线程同步问题导致Looper为null
> 2. 在getLooper()调用时，可能run()方法中Looper还没有创建成功，此时返回Looper会出现问题。因此需要去wait。
> 3. run()中Looper一旦创建好，就可以去notifyAll()让getLooper()可以返回正确的Looper

6、为什么要设置线程优先级？为什么能防止内存泄漏和解决数据安全问题?

> 1. 线程优先级主要分为UI线程和后台线程(backgorund)
> 2. 线程默认优先级为default，也就是UI线程，会和UI线程去平等的争抢资源。
> 3. 在UI性能要求较高的场景下，应该使用后台线程。

7、线程优先级设置的两种方法?两种方式的优先级范围?

> 1-setThreadPriority()进行设置，线程优先级范围-20~19, 从高优先级~低优先级。

```
android.os.Process.setThreadPriority (intpriority);
android.os.Process.setThreadPriority (inttid, int priority);
// thread直接设置
Thread a = new MyThread("ThreadA");
a.setOSPriority(Process.THREAD_PRIORITY_LOWEST);// 19
// thread的run方法中进行设置
Thread b = new Thread(new Runnable() {
    @Override
    public void run() {
        Process.setThreadPriority(Process.THREAD_PRIORITY_LOWEST);
    }
});
```

> 2-setPriority()设置，线程优先级范围1~10, 从低优先级~高优先级。

```
java.lang.Thread.setPriority (int priority);
// setPriority进行设置
Thread a = new MyThread("ThreadA");
a.setPriority(Thread.MAX_PRIORITY);// 10
```