Java 核心技术(进阶)

第五章 Java多线程和并发编程

第三节 Java多线程信息共享

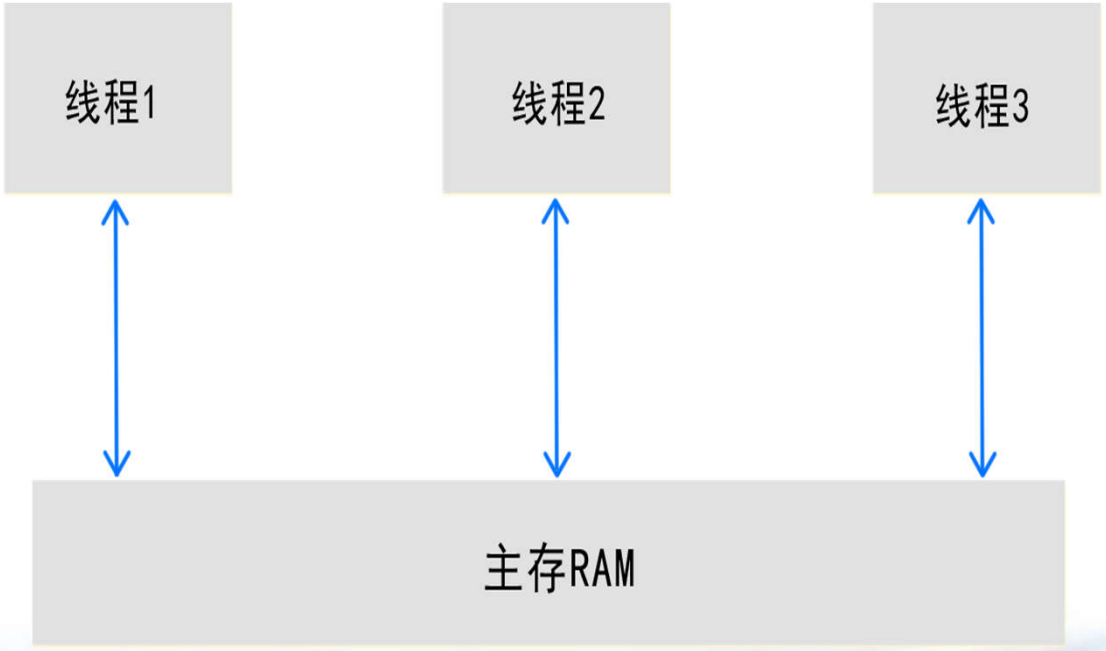
华东师范大学 陈良育

多线程信息共享(1)

- 线程类
  - 通过继承Thread或实现Runnable
  - 通过start方法，调用run方法， run方法工作
  - 线程run结束后，线程退出
- 粗粒度：子线程与子线程之间、和main线程之间缺乏交流
- 细粒度：线程之间有信息交流通讯
  - 通过共享变量达到信息共享
  - JDK原生库暂不支持发送消息(类似MPI并行库直接发送消息)

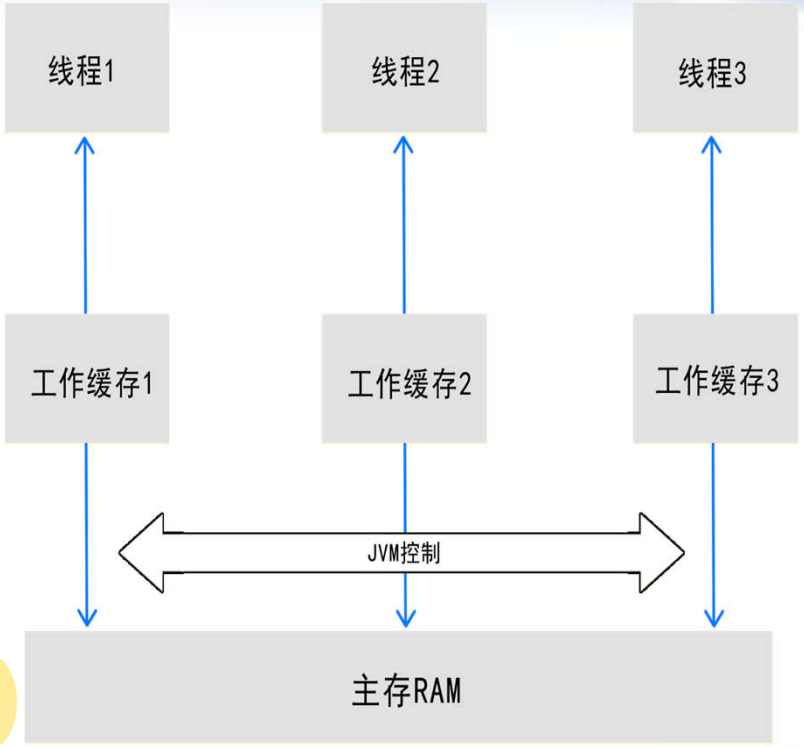
多线程信息共享(2)

- 通过共享变量在多个线程中共享消息
  - static变量
  - 同一个Runnable类的成员变量
- 查看示例



多线程信息共享(3)

- 多线程信息共享问题
  - 工作缓存副本
  - 关键步骤缺乏加锁限制
- i++，并非原子性操作
  - 读取主存i(正本)到工作缓存(副本)中
  - 每个CPU执行(副本)i+1操作
  - CPU将结果写入到缓存(副本)中
  - 数据从工作缓存(副本)刷到主存(正本)中



多线程信息共享(4)

- 变量副本问题的解决方法
  - 采用volatile 关键字修饰变量
  - 保证不同线程对共享变量操作时的可见性
- 查看示例

多线程信息共享(5)

- 关键步骤加锁限制
  - 互斥：某一个线程运行一个代码段(关键区)，其他线程不能同时运行这个代码段
  - 同步：多个线程的运行，必须按照某一种规定的先后顺序来运行
  - 互斥是同步的一种特例
- 互斥的关键字是synchronized
  - synchronized代码块/函数，只能一个线程进入
  - synchronized加大性能负担，但是使用简便
- 查看示例

总结

- 总结
  - 了解多线程之间信息共享机制
  - 掌握volatile和synchronized的用法

代码(1) ThreadDemo0.java


```
public class ThreadDemo0
{
    public static void main(String [] args)
    {
        new TestThread0().start();
        new TestThread0().start();
        new TestThread0().start();
        new TestThread0().start();
    }
}
```

代码(2) TestThread0.java


```
class TestThread0 extends Thread
{
    //private int tickets=100;              //每个线程卖100张，没有共享
    private static int tickets=100;   //static变量是共享的，所有的线程共享
    public void run()
    {
        while(true)
        {
            if(tickets>0)
            {
                System.out.println(Thread.currentThread().getName() +
                " is selling ticket " + tickets);
                tickets = tickets - 1;
            }
            else
            {
                break;
            }
        }
    }
}
```

代码(3) ThreadDemo1.java


```
public class ThreadDemo1
{
    public static void main(String [] args)
    {
        TestThread1 t=new TestThread1();
        new Thread(t).start();
        new Thread(t).start();
        new Thread(t).start();
        new Thread(t).start();
    }
}
```

代码(4) TestThread1.java


```
class TestThread1 implements Runnable
{
    private int tickets=100;
    public void run()
    {
        while(true)
        {
            if(tickets>0)
            {
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                tickets--;
                System.out.println(Thread.currentThread().getName() +" is selling ticket " + tickets);
            }
            else
            {
                break;
            }

        }
    }
}
```

代码(5) ThreadDemo2.java


```
public class ThreadDemo2
{
    public static void main(String args[]) throws Exception
    {
        TestThread2 t = new TestThread2();
        t.start();
        Thread.sleep(2000);
        t.flag = false;
        System.out.println("main thread is exiting");
    }
}
```

代码(6) TestThread2.java


```
class TestThread2 extends Thread
{
    //boolean flag = true;    //子线程不会停止
    volatile boolean flag = true;  //用volatile修饰的变量可以及时在各线程里面通知
    public void run()
    {
        int i=0;
        while(flag)
        {
            i++;
        }
        System.out.println("test thread3 is exiting");
    }
}
```

代码(7) ThreadDemo3.java


```
public class ThreadDemo3 {
    public static void main(String[] args) {
        TestThread3 t = new TestThread3();
        new Thread(t, "Thread-0").start();
        new Thread(t, "Thread-1").start();
        new Thread(t, "Thread-2").start();
        new Thread(t, "Thread-3").start();
    }
}
```

代码(8) TestThread3.java


```
class TestThread3 implements Runnable {
    private volatile int tickets = 100; // 多个 线程在共享的
    String str = new String("");

    public void run() {
        while (true) {
            sale();
            try {
                Thread.sleep(100);
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
            if (tickets <= 0) {
                break;
            }
        }

    }

    public synchronized void sale() { // 同步函数
        if (tickets > 0) {
            System.out.println(Thread.currentThread().getName() + " is saling ticket " + tickets--);
        }
    }
}
```

谢 谢！