

Very big topic -- 433 covers Java concurrency in great detail.

A concurrent program has more than one active "execution context" -- effectively the computer is asked to do more than one thing at a time. You are asking a computer to multitask.

One way to implement different execution contexts is to have different "threads of control."

Why concurrency?

Thursday, April 30, 2009
1:08 PM

1. Sometimes concurrency is the most natural way to describe a program.

Example: Managing multiple simultaneous requests to a shopping website

2. Interruptions happen.

Example: keyboard input.

3. To increase performance by getting more done in less time!!

Parallel vs quasiparallel

Thursday, April 30, 2009
1:13 PM

Parallel: doing two or more things at literally the same time.

Quasiparallel: switching between two or more tasks at unpredictable times

Programming principles for parallel and quasiparallel programming are the same.

Thread, process, etc. -- these are operating system issues. I personally use "threads" to mean execution contexts that share an address space and "processes" to mean execution contexts that each have their own address space, but the actual definition varies by operating system or run-time VM.

1. Communication: How do execution contexts share info?

- share memory
- pass messages

2. Synchronization: How can the programmer control the order in which operations occur in different execution contexts??

A programming language needs to provide a way for the programmer to express what is communicated and in what order operations should happen.

My best advice for concurrent programming

Thursday, April 30, 2009

1:22 PM

1. Don't have the execution contexts talk to each other. Silence!
2. Design it so that order of operations doesn't matter!!!!

How can we do this???

PURE FUNCTIONAL PROGRAMMING!

- no side effects, no I/O
- function can be called at any time, will give same output given same input

Anything that can be converted to functional programming style should be.

Sometimes you have to communicate or share. Bummer.

Thursday, April 30, 2009
1:24 PM

Programming language mechanisms to help:

1. Monitor

A class-like entity that ensures only one method is executed at a time. If you call a method of a busy monitor, you have to wait.

Java:

```
class RestroomStall {  
    synchronized void enterRestroomStall() {...}  
    synchronized void cleanRestroomStall() {...}  
}
```

2. Conditional critical sections

A syntactically delimited section of code that provides access to a protected variable.

Java:

```
synchronized (my_shared_obj) {  
    // code that accesses the shared  
    // object  
}
```

Only one synchronized statement that refers to a shared object can execute at one time.

3. Locks

Java:

```
Lock l = new ReentrantLock();  
/* reentrant lock means I can reacquire the  
the lock as many times as I want */  
l.lock();  
/* access all kinds of shared objects */  
l.unlock();
```

Back to functional programming...

Thursday, April 30, 2009
1:37 PM

If we have a bunch of code that doesn't have side-effects, the compiler can detect which parts of the code can be executed in parallel!

Implicit parallelism! I don't have to define separate threads at all!

Sisal: a functional programming language with such a compiler

Example of Sisal code: matrix multiplication

Tuesday, May 05, 2009
12:56 PM

Sisal compiler automatically generates parallel code. Sisal's for loop is only for loops where later iterations don't depend on values from previous iterations.

```
define main

type OneDim = array [ real ];
type TwoDim = array [ OneDim ];

% generates two matrices
function generate( n : integer
                  returns TwoDim, TwoDim )

    for i in 1, n cross j in 1, n
        returns array of real(i)/real(j)
        array of real(i)*real(j)
    end for
end function % generate

function mult( n : integer; A, B : TwoDim
              returns TwoDim )

    for i in 1, n cross
        j in 1, n
            c := for k in 1, n
                t := A[i,k] * B[k,j]
                returns value of sum t
            end for
        returns array of c
    end for
end function % mult

function main( n : integer returns TwoDim )
```

```
let A, B := generate( n )  
in mult( n, A, B )  
end let
```

```
end function % main
```

Pasted from <<http://www2.cmp.uea.ac.uk/~jrwg/Sisal/01.Introduction.html>>

Futures

Thursday, April 30, 2009

1:51 PM

`(future (my-func my-args))`

future arranges for the function to be called in its own execution context.

The caller of my-func continues to execute until it needs the return value of my-func, in which case it blocks.

Bummer: future not implemented in today's Scheme interpreters.

But -- you can use Futures in Java!

```
Future<Integer> result =  
executorService.submit( callable-object );  
int val = result.get(); //blocks until ready
```

Example of Java Futures

Tuesday, May 05, 2009
12:47 PM

```
interface ArchiveSearcher { String search(String target); }
class App {
    ExecutorService executor = ...
    ArchiveSearcher searcher = ...
    void showSearch(final String target) throws InterruptedException {
        Future<String> future = executor.submit(new Callable<String>() {
            public String call() { return searcher.search(target); }
        });
        displayOtherThings(); // do other things while searching
        try {
            displayText(future.get()); // use future
        } catch (ExecutionException ex) { cleanup(); return; }
    }
}
```

Pasted from

<http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/Future.html>

Two versions of Future<V>.get

Tuesday, May 05, 2009
12:48 PM

∨

get()

Waits (blocks) if needed for computation in separate execution context to complete, and then retrieves its result.

∨

get(long timeout, TimeUnit unit)

Waits (blocks) for a specific amount of time. If return value isn't ready, it throws a `TimeoutException`.

Pasted from <<http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/Future.html>>