

CSSE4004/7014
The University of Queensland

Tutorial 2
Notes on Solutions

1. Sketch out in pseudocode the algorithms for a server using the three given variations in implementation strategy, and in each case, comment on advantages and disadvantages:
 - a. Multiple threads using blocking system calls.

The solution is conceptually simple. We need a thread for each time we want to do a system call. There are several alternative ways to structure the solution. You could have a fixed number of threads, and when a system call is required, a thread from the pool is woken up and asked to do the call. Here, a simpler approach is illustrated. Each time a system call is required, a new thread is created, and terminates when the call completes. Why have I parametrised the call handler thread, rather than put in a specific call type and completion action?

```
thread handleCall (callName, completionAction)
    result = systemCall (callname);
    do(completionAction, result);
end thread

thread dispatcher
    while true
        on IOevent:
            determine nature of IO
            select systemCall
            select eventHandler
            launchThread (handleCall (systemCall, eventHandler)
        end while
end thread
```

- b. Single thread using non-blocking system calls.

The solution is more complicated to understand. You need to pass the nonblocking system call a completion routine to invoke when the I/O completes. Making this work correctly is nontrivial, because it will be invoked during an interrupt, when the state of your code may be hard to predict. The approach here is to allocate data to the completion routine, and to poll a location in that data to see if it has run to completion. It is assumed that the system call can be passed the location where the completion routine would like to store

its result, as well as the identity of the completion routine. To probe further: look up how asynchronous I/O is handled in your favourite OS.

```
IOcompletionHandler (data, IOresult)
    data.result = IOresult;
    data.finished = true;
end IOcompletionHandler

main program
    while true
    on IOevent:
        determine nature of IO
        select systemCall
        allocate buffer
        buffer.finished = false
        select eventHandler
        systemCall (eventHandler, data)
        foreach buffer
            if (buffer.finished)
                process buffer.result
                deallocate buffer
            end if
        end foreach
    end while
end main program
```

c. Finite-state machine, using nonblocking system calls.

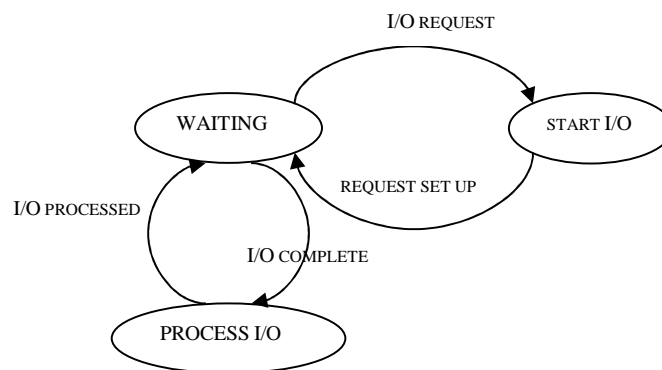


Figure 1. Finite-state machine, using nonblocking system calls.

In Figure 1, the general idea is illustrated. The simplest implementation would be a loop representing the "waiting" state, which on receiving an I/O request (could be through

several means -- an interrupt, it could poll a variable which is changed when an I/O request arrives) would call a start I/O function, set up the request and return to the waiting loop; other transitions would be the same. An FSM can also be represented as a table in 2 dimensions. Each row represents the transitions out of a particular state. Entries in the row correspond to each event. For example, if "waiting" is state 0, "start I/O" is state 1 and "process I/O" is state 2, the first row of the table would be as follows (filling in the missing entries should not be much of a challenge):

I/O Request	Request set up	I/O Complete	I/O Processed
0	1	-	2
1			
2			

2. One of the key issues in the viability of process migration is whether the gain (moving computation where it's more efficient) outweighs the cost (time to ship the code and any state which may be needed). In each of the following, estimate the likely cost versus gain:

The calculations here are quite simple so I will not work through all of them in detail. If answering a question like this for assessment, show all working, in case you have a different but valid interpretation of the question.

- a. A web server, with a total of 20Mbytes of address space, is loading a machine too much, and you would like to move it to another which can access the same files. In about 10 minutes, other users of the machine it's running on will have finished, but their processes are very large and even less practical to move. Your network has a bandwidth of 100Mbit/s, and the web server is continuously active, so you will need to move the entire state, then update anything which has changed at the end of the move.

Calculate the time to move the 20Mbytes = 160Mbits at 100Mbit/s -- 1.6s. Compare with the 10 minutes before the machine is less loaded: it does seem worthwhile. However, if you are going to have to build a mechanism for logging web server activities to handle the last part of the question, you need to think about whether doing so will add sufficiently to average run time to make it preferable not to migrate the server. Alternatively, you could tell the person spec'ing the problem that they need to consider whether taking the server down for under 2s is really a problem. In the real world, the spec is more mutable than in an academic question: it is good professional practise if you are asked to do something which doesn't make sense to explain why a more realistic requirement will meet the real needs, cost less, etc.

- b. A picture is drawn as part of a web user interface. The code which draws it, in Java, is 20Kbytes. The data for the picture is 70Kbytes, and the finished picture is 80Kbytes.

Basic calculation: data + code = 90Kbytes.

- i. If it is unlikely to be drawn more than once for a given client, does it make sense to ship the Java code over, rather than just sending the picture? Explain if there are situations where it may make sense, and others where it does not.

Clearly on the numbers alone, you wouldn't ship it – but what could make the picture change dependent on where it's drawn, which would not be known on the server side?

- ii. How would your answer change, if you knew that the picture would be drawn more than once? If you knew that slight variations on the picture could be drawn, as the user interacted with the web page?

Straightforward.

3. In the X Window system, explain which of the following components plays the role of a client, and which plays the role of a server:
 - a. The machine on your desktop launches a large computation on a machine with high-speed computation capability.

Obvious.

- b. The machine with high-speed computation capability asks the X software on your desktop to display some output.

Your desktop machine here is an X server. Why? It provides the service of displaying output. The other machine is the client.

- c. The machine on your desktop runs a window manager on another machine in your previous office.

Should be obvious if you understand 4b.

4. Is a server that maintains a TCP connection to a client stateful or stateless?

The server is stateless as it does not maintain information about clients. The transport layer maintains the state, not the server.

5. A web server maintains a table in which client IP addresses are mapped to the most recently accessed Web pages. When a client connects to the server, the server looks up the client in its table, and if found, returns the registered page. Is this server stateful or stateless?

The information that the server keeps is not needed to maintain correct interactions between the server and its clients. Therefore it can be argued that the server is stateless.

6. Discuss why code migration across heterogeneous systems with programs written in C++ is a hard problem. List issues which are difficult to solve, and potential solutions.

Problems include potentially different architectures running different instruction sets, different data representations on different machines and the fact that memory layout will likely differ so pointers cannot be used without action to keep them consistent across machines. Solutions include running on a virtual machine like Java's and maintaining a machine-independent representation of the call stack and only allowing migration at a call. There's a reasonably detailed discussion of the options in Tanenbaum and Van Steen, pp 110-112. Make sure you understand all the issues raised.