Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.826 – Principles of Computer Systems

Handout ?   October 20, 1993

Problem Set #6

Due date: October 27, 1993

Problem 1 (? points)

We want to implement buffered output streams that are automatically flushed by a background thread. We will call these streams "writers". The output to a writer goes from its buffer into something called a "sink", the exact nature of which is unimportant for this problem—it might be a terminal or a file, for example. Here is the spec:

```
MODULE Wr

TYPE Sink     = Int
     Wr       = Int                              % index into w

VAR  contents : Sink -> SEQ Char := {* -> {}}
     w        : SEQ [sink: Sink, buf: SEQ Char] := {}

PROC Flush(wr) = <<
    contents(w(wr).sink) := contents(w(wr).sink) + w(wr).buf
    w(wr).buf := {} >>

PROC New(sink) -> Wr = <<
    w := w ++ {sink := sink, buf := {}}; RET w.size-1 >>

PROC PutChar(wr, char) = << w(wr).buf := w(wr).buf ++ char >>

THREAD Worker() = DO <<
    VAR wr | w!wr /\ w(wr).buf # {} => Flush(wr)
    [*] SKIP >> OD

END Wr
```

(A) (? points)

Implement the procedures in this module except for Flush. Treat Flush as a black box provided by someone else (but your program must provide mutual exclusion for the flush action). Your solution should satisfy the following requirements:

- Different threads can call PutChar concurrently on different writers and not delay each other.
- No character should remain unflushed in a buffer for more than 50 ms or so. To meet this requirement, invent a suitable Delay primitive. You may assume that everything except Delay executes in an amount of time that is small compared to 50 ms.

It should be obvious how to translate your implementation into a reasonable program in C, Clu, or whatever: there should be no spin locks or repeated examination of writers that don't need to be flushed, writers should require only bounded buffers, it should be clear how to implement any sets as linked lists, etc.

Explain informally why your program works; be sure to give the invariants it maintains.

Hint: The crucial issue is what mutexes to have and in what order to acquire them. The program should not be more than a couple of pages.

(B) (? points)

How would your program differ if it acquired the mutexes in the opposite order?

(C) (? points)

Discuss what the impact would be on your program if a sink might contain a writer inside it, and flushing to the sink could cause PutChars to that writer.