From: Dawson Engler <engler@csl.Stanford.EDU>
Subject: Re: Quals Question
To: shankle@ee.stanford.edu (Diane Shankle)
Date: Wed, 9 Feb 2000 16:42:24 -0800 (PST)

Hi Diane,

> I am still missing your Quals Question! Please try to get it in this week.!

here it is; good luck extracting one from the other poeple ;-)

Dawson

Dr. Frankenstein (Jr.) has decided that people are too stupid to deal with concurrency and has started a company (``not-like-last-time.com'') that plans to genetically modify the population to correct this problem.

As a stopgap measure while he works out a few details in the labratory, he has hired you to eliminate the need for locks by building a system he has designed that works as follows:

1. On thread create, it creates a private copy of the current (``primary'') address space. Every load or store done by the thread uses this private copy.

2. When a thread exits, the system attempts to merge the thread's private address space back with the primary copy. If the system detects there was a possible a race condition, it discards the private address space and restarts the thread from scratch.

   To do this step, the system allows you to suspend all threads, and to later resume them.

3. To allow you to detect races, the system tracks (1) the time the thread was created, (2) what pages have been read or written by the process and (3) every primary memory page has a time stamp of the last time it was read or written.

Explain how to use the system's provided information to detect when a possible race condition happened. Please be concretely precise about what a race condition will look like; and say each of the four possible combinations of read and write why it is ok or causes a race.

We rely on being able to reexecute threads. What assumptions are we making? (Hint, think about restarting instructions after a fault.) Give two concrete examples of things the thread cannot do.

System status might not be same.
IO or network packet may lost