Student ID: ___________

CS-736 Final: To Err is Human
(Fall 2003)

An Exam offered to you by Professor Remzi Arpaci-Dusseau

Please Read (And Answer) All Questions Carefully!

There are twelve (12) total numbered pages; there are 20 equally-weighted questions.

Please put your NAME on this page, and your STUDENT ID on this and all other pages

Name: ____

Introduction

Welcome to an alternate universe!

All semester long, we've talked about systems that have been built throughout the history of systems research. All of these systems have both flaws as well as numerous qualities that make them worth studying. Of course, not every aspect of each system is the best or only way to build said system; indeed, it is conceivable to think of many other variants of each.

In this exam, we present slight variants of each of the systems we have studied; assuming the system we studied is called $SuperSystem$, we will call the variant introduced in this exam $SuperSystem_{Bizarro}$. Your job will be to answer questions about these alternate systems, as described in more detail below in each question. Good luck!

1. The designers of $Disco_{Bizarro}$ wanted it to run on a new machine from SGI, called the $Origin_{Bizarro}$. This machine was the same as the machine described in the Disco paper, except that it only had *two* processor modes: user mode (restricted) and kernel mode (full privileges). **Do you think the designers of $Disco_{Bizarro}$ will be able to implement their system efficiently? Discuss.**

2. In $FFS_{Bizarro}$, there is a feature that tries to take advantage of temporal locality. Specifically, when two directories are created at nearly the same time, $FFS_{Bizarro}$ tries to place them within the same cylinder group. **Compare and contrast FFS and $FFS_{Bizarro}$ allocation. What are the strengths and weaknesses of each approach?**

3. $AFS_{Bizarro}$ is quite similar to AFS, with this key difference. Instead of moving whole files to the client upon open, $AFS_{Bizarro}$ only moves the first 64KB. Other 64KB chunks are moved to the client lazily, on demand. **Compare and contrast AFS and $AFS_{Bizarro}$, both in terms of performance and consistency.**

4. In $Gray - box_{Bizarro}$, a different method for determining the contents of the OS file cache is used. Specifically, a library is used to monitor read and write requests, and the OS cache contents are derived via a simulation. For example, if LRU is the underlying policy, an LRU list is maintained in the library; a read of a page would move it to the front of the simulated LRU list, as it does in the kernel file cache. Applications can then look at this user-level LRU list to determine what is in the cache. **Compare this approach to the probe-based approach taken in the original gray-box system. What are the strengths of each approach, and what are the weaknesses?**

5. The design of $Multics_{Bizarro}$ included all of the same support for segmentation as in $Multics$; however, $Multics_{Bizarro}$ does **not** include any support for paging. **Discuss the strengths and weaknesses of the $Multics_{Bizarro}$ design.**

6. In the paper we read for class, $Baker$ observes all *deletions* that have occured within the trace sample (of say 24 hours), finds the creation time of the file, and thereby can compute average file lifetime for all files that have been deleted. In a new measurement paper, a different method of measuring file lifetime is introduced. Specifically, $Baker_{Bizarro}$ decides to observe all file *creations* that have occured within a trace sample, and then compute the average lifetime for those files; specifically, if a file is deleted during the sample, the exact lifetime goes into the average, and if not, the lifetime is assumed to be the length of the sample. **Compare and contrast $Baker$ and $Baker_{Bizarro}$; which is the better technique for computing average file lifetime?**

7. When issuing a long duration call in the Xerox RPC system, the client repeatedly probes the remote server, getting acknowledgments from it, until the result of the call finally is returned. $RPC_{Bizarro}$ does not do this; instead, the server will send a single ack that it has received the call and is working on it, and of course an ack in the form of the final result. **What is the problem with the $RPC_{Bizarro}$ approach?**

8. $LFS_{Bizarro}$ is a variant of LFS that streamlines the contents of a segment. In fact, $LFS_{Bizarro}$ does not even include a *segment summary block* within each segment, just inodes, pieces of the imap, and data blocks. **What aspect of LFS *cleaning* is made more difficult without a segment summary block?** (Just focus on cleaning, and not other aspects of LFS that may also need to change.)

9. In $Exokernel_{Bizarro}$, there is an additional feature that lets user-level “OS libraries” download code into the OS, specifically for the scheduler. A library can download scheduler code which changes the default multiplexing employed by exokernel. **Why would such a feature be useful? What are the inherent difficulties in providing such a feature?**

10. $Mesa$ is a concurrent programming environment, with monitors and condition variables used to build parallel programs and even the Pilot operating system. One feature included in $Mesa$ is called the “naked notify”. This type of notify, as you may recall, can signal a condition but does so without grabbing the monitor lock, and required changes in other parts of the $Mesa$ code to support it. Our alternate version of $Mesa$, $Mesa_{Bizarro}$, includes a naked notify, but does not include those changes. **What problems can a naked notify lead to? How are these problems avoided in the original $Mesa$ system?**

11. Resource Containers (RC) are used to charge resource consumption within a system on a per-activity basis. The slight variant, $RC_{Bizarro}$, also provides a flexible mechanism for accounting, but only accounts for *kernel resources* that are consumed. **In an event-based web server, is the functionality provided by $RC_{Bizarro}$ "good enough" to deliver differentiated services to clients?**

12. Scheduler Activations (SA) are used to inform the user level of kernel activity that is relevant to user-level thread management. Specifically, four upcalls are provided: *Add this processor*, *Processor has been preempted*, *Scheduler activation has blocked*, *Scheduler activation has unblocked*. The designers of $SA_{Bizarro}$ thought this was one too many upcalls, and decided to remove one. Specifically, they removed *Scheduler activation has unblocked.* **Is this a good decision? Why or why not?**

13. The Vax/VMS paging system uses first-in-first-out (FIFO) queues to manage pages in memory. $Vax/VMS_{Bizarro}$ instead uses last-in-first-out (LIFO) queues. **Discuss how the original Vax/VMS system uses FIFO queues to approximate LRU, and how the $Vax/VMS_{Bizarro}$ variant would behave.**

14. In $Nucleus_{Bizarro}$, the designers forgot to include an external clock process. Recall that a process could send a message to the clock process, and then receive an answer back some time later, as specified by the sending process. **Why is an external clock process useful in Nucleus, and what are the implications of not having such a process in $Nucleus_{Bizarro}$?**

15. The THE system is composed of many layers. $THE_{Bizarro}$ takes a similar approach, but switches the lower two layers. Specifically, instead of virtualizing the CPU at layer 0 and virtual memory at layer 1, $THE_{Bizarro}$ virtualizes virtual memory at layer 0 and the CPU at layer 1. **Does the order of virtualization of $THE_{Bizarro}$ make sense? Discuss.**

16. During reintegration in $Coda$, conflicts are detected and when not automatically fixable, given to the user so that human intelligence can be applied to the problem. In the design of $Coda_{Bizarro}$, one of the main goals is to detect and repair all conflicts automatically. **Why is this difficult?** (giving examples is a good way to answer this problem)

17. In NFS, the designers' goal was to make as many operations idempotent as possible. The designers of $NFS_{Bizarro}$ thought idempotency was overrated, and instead introduced some non-idempotent variants of the protocol. Specifically, they added a *write* call that took the following arguments: *file handle, data, size*. **What problem does this non-idempotent variant of NFS introduce?**

18. In this question, we discuss $RAID - 1_{Bizarro}$, a variant of RAID Level 1 (Mirroring). Assume that in pure mirroring, for each disk D, there is one other disk D' that contains the exact same contents as D. $RAID - 1_{Bizarro}$ works slightly differently. Specifically, for each chunk of blocks, $RAID - 1_{Bizarro}$ keeps one *vertical* copy (on the same disk), and one *horizontal* copy (across disks), of course making sure that a block and its copy are on separate disks. For example, in a 4-disk system, we could have blocks 0, 1, and 2 on disk 0, and then spread blocks 0', 1', and 2' across the other 3 disks. Compare the **performance** and **reliability** of RAID-1 versus $RAID - 1_{Bizarro}$.

19. $Hydra_{Bizarro}$ is identical to $Hydra$, except in one aspect: there is no "walk" right. **What are the ramifications of not having a "walk" right in $Hydra_{Bizarro}$?**

20. In a new authentication system using private keys, a slight variant of the protocol discussed in the Needham and Schroeder ($N\&S$) paper is introduced. Let's call the new protocol $N\&S_{Bizarro}$. $N\&S_{Bizarro}$ is a private key protocol, with message exchange as shown below. **What is wrong with the $N\&S_{Bizarro}$ protocol?**

    (a) $A \rightarrow AS :$ A, B, I_{A1}

    (b) $AS \rightarrow A :$ $\{I_{A1}, B, CK, \{CK, B\}^{KB}\}^{KA}$

    (c) $A \rightarrow B :$ $\{CK, B\}^{KB}$

    (d) $B \rightarrow A :$ $\{I_B\}^{CK}$

    (e) $A \rightarrow B :$ $\{I_B - 1\}^{CK}$