Computer Sciences @ Wisconsin – OS Qualifier: Fall 2011

Instructions: Answer all six questions.

1. Fault, Error, Failure

a) What is the definition of fault, error, and failure?

b) Provide a concrete example of fault in a software system. Explain how to handle this fault before it turns into an error.

c) Provide a concrete example of error in a software system. Explain how to handle this error before it turns into an failure.

d) Provide a concrete example of a sub-system failure in a distributed system. Explain how to handle this sub-system failure and prevent full-system failures.

2. Security and virtualization:

Virtualization and virtual machine monitors have often been used to improve the security of a system, as in the VAX VMM security kernel.

a) Both VMware and the VAX security kernel use ring compression to implement execute the virtual machine monitor at the highest privilege level. What is ring compression, and what are its effects on software running within the virtual machine?

b) What additional features does a secure VMM require as compared to a normal virtual machine monitor, and why?

c) For the following two scenarios, explain whether implementing security within a VMM offers benefits as compared to implementing it within the operating system, and why.

   i. A user on the system is trying leak classified data.
   ii. An external agent is trying to break into the system over a network.

3. RAID

Assume you have a RAID-4 system with D+1 disks, with 1 disk used for redundancy information (the rest for data). Each disk has B blocks of capacity, and all disks are of identical make, model, and performance. Unfortunately, one of your data disks fails and will not work anymore, and you must replace it with a brand new disk. The system must then carry out the task of updating the new disk so that the RAID-4 returns to its normal "working" status (i.e., all the data disks have all their data and the parity disk has all its necessary parity information), a process known as reconstruction.

a) Describe how the process of reconstruction would work for RAID-4. How does the process change if a data disk or parity disk fails?

b) During this reconstruction, how many blocks must be read from the RAID-4 storage system, and how many written?

c) Now assume that we had instead created a storage system with mirroring instead of RAID-4, with a total of 2D disks for the system. Describe how reconstruction would work in the mirrored system.

d) Which is faster - reconstruction in RAID-4 or reconstruction in the mirrored system? Justify your answer.

4. Memory Management

Memory management systems can have a strong impact upon system performance. In this question, we explore how memory management should evolve on some interesting hardware configurations. In particular, we focus on hardware systems that support large page sizes; assume in this case we have a machine that in addition to a standard 4 KB page allows for large 1 MB pages.

a) Describe the motivation for large pages. Why should hardware systems provide this kind of support?

b) Let's say we want to provide an application interface to allow applications to ask for large pages. (i) How does an OS typically hand out memory, and (ii) how should the system call and library API be enhanced to include the ability to request small or large pages?

c) Now let's say we want to get the benefits of large pages without changing applications or the memory-request interface. Describe how an OS could transparently use large pages, and how it would do so.

5. Threads

a) Consider a user-level thread library intended to run on a modern Unix type system (such as Linux). Assume that a multi-threaded program using this user-level library is running on a single Linux process.

   What are the roles of the thread library and the kernel in operations such as thread creation, synchronization and blocking between these threads, and I/O operations? Use a diagram to illustrate your answer.

b) Modern UNIX type systems support processes that share the same address space (often called lightweight processes (LWP's) or kernel threads). In such systems, there are still user-level thread libraries, but these libraries will create one LWP per user thread.

   What are the roles of the thread library and the kernel in operations such as thread creation, synchronization and blocking between these threads, and I/O operations? Again, a diagram will help your answer.

c) Most current operating systems (including Linux and Windows) have a fixed binding of a thread to a specific LWP (so, a one-to-one correspondance of threads to LWP's) Some past versions of Unix allow a program to have any number of threads and LWP's, with any thread being able to run on any LWP. In these systems, there is not a fixed assignment or one-to-one correspondence of user-level threads to the LWP's; a collection of T user threads can be scheduled on any of the L LWP's.

   What are advantages and disadvantages of this design over the one-to-one assignment of threads to processes?

6. Operating Systems for 64-bit Machines:

Most current processors have 64-bit virtual addresses. It has been suggested that such a huge virtual address space should be managed in a new way. In "traditional" systems, each process is given its own address space. Sharing is supported (if at all) by mapping pages of physical memory into virtual address spaces of two or more processes, perhaps at different virtual addresses. A flat scheme, by contrast, would have one huge virtual address space shared by all processes. Each process would be authorized to access only certain portions of the space.

What advantages does the flat scheme offer over the traditional approach? What new problems arise?