CSE473/Spring 2008 - 1st Midterm Exam

Tuesday, February 19, 2007 — Professor Trent Jaeger

Please read the instructions and questions carefully. You will be graded for clarity and correctness. You have 75 minutes to complete this exam, so focus on those questions whose subject matter you know well. Write legibly and check your answers before handing it in.

Short Answer - some will be one or two words – no more than 3 sentences

1. (*3pts*) Name two types of traps generated by user programs on purpose to further their execution. Name one trap that is caused by a user program error.

   answer: Further: System call and page fault. Error: Divide-by-0 and memory fault.

2. (*4pts*) What is the difference between the 1:1 and M:1 threading models? Why do we prefer a 1:1 threading model?

   answer: These models define the number of user-space threads per kernel thread. 1:1 has one kernel thread for each user-space thread. M:1 has all the user-space threads on one kernel thread. The 1:1 model permits a user-space thread to block while allowing other user-space threads to run since each corresponds to a schedulable kernel thread.

3. (*4pts*) What function does a processor cache provide (be specific)? Why is *caching* often used in operating system (think generally)?

   answer: A processor cache stores the result of a memory load and subsequent writes to that memory location in the processor, so the processor does not have to submit a memory request on subsequent accesses. Caching in operating systems avoids unnecessary recomputation by storing the results, so they can be retrieved efficiently.

4. (*3pts*) What is the difference between *synchronous* and *asynchronous* message passing IPC?

   answer: In synchronous MP IPC, the sender and receiver must both be ready to communicate in order for the message to transferred (so one must wait). In asynchronous MP IPC, a third party (kernel or "mailbox") holds the sent message for the receiver to retrieve when they are ready.

5. (*3pts*) What does it mean for a scheduler to be *preemptive*?

   answer: A preemptive scheduler can reschedule (transition to ready) a running task even when it is does not yield the CPU (e.g., by being blocked).

6. (*3pts*) Under what condition is *shortest job first* scheduling optimal with respect to minimizing the average waiting time? Explain briefly.

   answer: SJF is optimal if we know the length of the next CPU burst of all ready processes and it is preemptive (either or both are acceptable). It then orders the processes from least to greatest CPU burst at all times thus minimizing the average waiting time of all.

7. (*3pts*) What performance costs are saved by the use of thread pools?

   answer: Thread creation, termination, and context switching costs. Note that we can reuse the same thread over again on multiple tasks.

8. (*4pts*) How does the library function `open`'s processing differ from that of the system call `open`?

   answer: The library function `open` runs in the address space of the user program and prepares the invocation of the system call (`int 0x80`). The `open` system call locates the corresponding file and creates a reference to access the file (if authorized).

9. (*3pts*) Which are shared among threads in the same process: (1) stack; (2) data segment; (3) heap?

   answer: Data segment and heap are shared.

10. (*3pts*) When a process is schedulable, what process state is it in? When a running process blocks, what state does it transition to? When an interrupt is taken, what state does a running process transition to?

    answer: ready, waiting, ready.

11. (*3pts*) What does a *scheduler activation* enable user programs to do?

    answer: A scheduler activation makes an upcall to a user program to enable the user program to choose whether to add a new thread before its last thread blocks.

12. (*4pts*) What is the main advantage of using a *microkernel architecture* for your OS? How can you use the microkernel architecture to optimize performance?

    answer: Main benefit is ease of extending the OS. Can improve performance by extending the OS with an optimized server (e.g., file server) – a program that is tuned for the applications on that system.

Long Answer - no more than 2 paragraphs

13. (*7pts*) How do you prepare a shared memory IPC mechanism (in general)? How does a sender use shared memory to send a message? How does a receiver receive a message via shared memory? Your description must ensure the integrity of reads and writes (you may ignore wrap-around).

    answer: The sender and receiver each agree to attach a common shared memory region into their address space. They also share references (pointers) to the last data read by the receiver and the last data written by the sender. When the sender writes a message, she adds the data from the last location written and updates this reference (after writing). When the receiver reads a message, she reads from the last location read to the last location written and updates this reference (after reading).

14. (*7pts*) Define the purpose of the TLB hardware. How do thread context switches make better use of the (non-tagged) TLB than process context switches? How is TLB usage degraded by multithreaded processes?

    answer: The TLB provides a mapping of virtual to physical addresses for an address space. Multi-threaded processes improve TLB usage because all the threads in a process have the same virtual-physical mapping, so the TLB entries still apply – no flush or refill is necessary on thread switch. Finally, threads impact TLB usage negatively because they have multiple stacks and some thread-local data that will take extra virtual-physical entries perhaps causing thrashing in the TLB.

15. (*7pts*) How is a dynamically-linked library different than a statically-link one? What is the memory-usage advantage of dynamically linked libraries over statically-linked ones? What is the advantage of using a *global offset table* to store function addresses rather than directly addresses library functions?

    answer: A dynamically-linked library is in a separate file from the executable, so it can be loaded on demand by the dynamic linker. In this way, dynamic linking saves on physical memory usage because the same library can be mapped into multiple address spaces, rather than being part of the executable's code segment. Finally, the GOT permits more effective use of virtual memory because each process can map the library into their own, independent location in its address space. Otherwise, all processes would need to agree on the location of the library in virtual memory.

16. (*7pts*) How does multilevel queue scheduling ensure that I/O-bound processes are given better scheduling than CPU-bound processes?

    answer: Multilevel queue scheduling defines scheduling criteria based on quanta limits that promote processes that do not use their quanta. A process initially is scheduled in a low priority queue, but has a long quanta when it runs. However, if the process does not use its quanta, it is promoted to a higher priority queue with a shorter quanta. This process is repeated.

Word Problems - take your time and answer clearly and completely.

17. (*12pts*) Consider the following set of processes, their CPU burst times, their arrival times, and their priorities (where a higher priority is better).

Process ID	CPU Burst Time	Arrival Time	Priority
P_1	6	0	2
P_2	3	4	3
P_3	4	5	1
P_4	1	7	6

    (a) (2pts) Draw a Gantt Chart (timeline) for the task schedule that would be generated using a First-Come, First-Serve scheduler.

(b) (2pts) Draw a Gantt Chart (timeline) for the task schedule that would be generated using a preemptive Shortest Job First scheduler.

(c) (2pts) Draw a Gantt Chart (timeline) for the task schedule that would be generated using a preemptive priority scheduler.

(d) (2pts) At what times would preemptive scheduling decisions be made by scheduling given this table.

(e) (3pts) Suppose the time quanta is 3 time units. Also, we use a two-level preemptive scheduler, where tasks are started in a higher priority level, but dropped to the second, lower priority level when they use their full time quanta (assume SJF within the level).

Draw a Gantt Chart (timeline) for the task schedule that would be generated using a preemptive priority scheduler.

(f) Using expotential average, what will the next estimate of CPU burst time be for P_1 assuming that $t_n = 6$ (real burst is same as estimated), $\alpha = 1$ and $\tau_0 = 3$?

answer: (a) In order.
(b) P1 (0-6), P2 (6-7), P4 (7-8), P2(8-10), P3(10-14)
(c) P1 (0-4), P2 (4-7), P4 (7-8), P1(8-10), P3(10-14)
(d) When processes arrive (4, 5, 7).
(e) P1 (0-4), P2 (4-7), P4 (7-8), P3 (8-12), P1(12-14)
(f) 6. Same as the t_n.

18. (*10pts*) Answer the following questions about creating/using a new process or thread.

(a) (2pts) What system call always creates a new address space in a UNIX system?

(b) (2pts) What is the difference between the `exec` system call and Linux `clone` with respect to where execution starts after the system call?

(c) (2pts) Can you use `exec` to run code in a different address space (i.e., different than the one from which `exec` was run)?

(d) (2pts) In a 1:1 threading model, what happens to the other threads when one thread makes a blocking system call?

(e) (2pts) Linux uses tasks to represent a thread of execution. Can two Linux tasks share a single address space?

answer: (a) fork
(b) exec starts at main always, but you can specify where you want to start from clone
(c) No
(d) They may be scheduled
(e) Yes

19. (*10pts*) Answer the following questions about a remote procedure call (RPC) implemented using a message passing IPC mechanism.

(a) (2pts) A procedure call to `server.help(int x, char *y)` is actually an RPC. A *stub* procedure in the client program finds the server process. What mechanism is used to package the RPC request (function name and arguments) for submitted as an IPC to the server?

(b) (2pts) Suppose the RPC is implemented by a *synchronous IPC* mechanism. Describe what happens if the client is ready to send before the server is ready to receive the IPC?

(c) (2pts) Suppose the RPC is implemented by an *asynchronous IPC* mechanism. What additional service must be provided by the system to implement asynchrony?

(c) (4pts) Suppose the RPC is implemented by an *asynchronous IPC* mechanism. Describe the mechanism by which a message is sent by the client and retrieved by the server. Identify any places where blocking is possible.

answer: (a) marshalling
(b) client blocks
(c) mailbox or message queue
(d) client contacts the mailbox. Client may block waiting for mailbox (briefly). Client sends message to mailbox (will wait for reply from server in RPC). Server contacts the mailbox for messages. Server may block for mailbox (briefly). Server will get message (if there) or continue (no blocking on server).