CS354 Final Solution, spring 2023

P1(a) 20 pts

Two processes P1, P2, and two semaphores S1, S2. P1 runs first and acquires S1. It is context-switched out. P2 runs and acquires S2. It tries to acquire S1 but blocks (and is context-switched out) since P1 has not released S1. P1 runs, tries to acquire S2 but blocks since P2 has not released S2.
8 pts

Kernel can maintain resource graph where a directed edge/link exists from a semaphore S to a process P, S -> P, if P has acquired S. A directed edge exists from P to S, P -> S, if P wants to acquire semaphore. A deadlock arises if there is a cycle in the resource graph which the kernel can check.
5 pts
// It's ok to omit definition of resource graph.

Cycle detection incurs linear overhead, and adverse effect of deadlock impacts (for the most part) deadlocked processes that comprise an app. Hence isolation/protection is not compromised, and high overhead justifies not providing deadlock detection service.
4 pts

Fix an order of all semaphores (i.e., total or linear ordering). All processes must acquire semaphores in the same order.
// In the above example, say order is S2 preceding S1. Even though P1 first needs
// S1, it must acquire S2 first, then S1. Hence even if P1 is context-switched
// out it cannot happen that P2 acquires S2 successfully since P1 has acquired it.
3 pts

P1(b) 20 pts

DMA intercepts and hides interrupts from CPU so that interrupt load is reduced.
DMA performs copy of data from hardware buffer to kernel memory.
8 pts

How much data (i.e., small data packets) to copy.
Where to copy the data.
6 pts

Producer: bottom half of lower half. In context-borrowing implementation of bottom half, the current process executes code of the bottom half. In kernel process/thread implementation of bottom half, a kernel process that executes bottom half is the writer/producer.
4 pts
// Assign half of the points if which process component is missing.

Consumer: upper half. The process that made system calls executes code of the upper half.
2 pts
// Same as producer if identifying which process is the consumer is missing.

P2(a) 20 pts

Main similarity:
Linear indexing overhead.
3 pts

Main difference:
XINU does not support hierarchical directory structure (i.e., 1-level flat structure withou subdirectories).
3 pts
// If other similarities/differences excluding the main ones are listed,
// assign 1 or 2 pts of partial credit.

Linear indexing structure is not suited for real-world files where most files are small ("mice") and a minority are large ("elephants").
2 pts

FAT file systems are typically used in mobile storage media such as USB flash drives (i.e. thumb drives) which are used mostly to transport a few large files. The components of large files tend to be located in contiguous areas of the physical storage medium which mitigates the overhead of linear indexing.

// For small files, if there are not many, overhead due to indexing remains
// low.
3 pts

File systems of commodity kernels must efficiently handle many small files and
a few large files which characterize real-world environments. Linear indexing
is not suited for such environments.
3 pts

UNIX/traditional file systems use a fixed number of direct pointers to handle
small files, and a tree structure to handle indexing of large files.
2 pts

This results in constant indexing overhead of small files, logarithmic overhead
of large files.
4 pts

P2(b) 20 pts

In a tickful kernel there is a fixed tick value (such as 1 msec on the Galileo
backends) at which the system timer is programmed to generate an interrupt.
4 pts

In XINU both upper half and lower are designed to disable external interrupts,
including system timer/clock interrupts, when executing. Since the wall clock
timer is a counter updated by the lower half (e.g., clkhandler() in XINU), during
kernel code execution XINU does not update the wall clock timer. This results
in inaccurate maintenance of wall clock time.
4 pts

XINU's upper and lower halves may be designed to not disable clock/external
interrupts and use semaphores to protect shared kernel data structures.
4 pts

In a tickless kernel there is no fixed tick value at which an interrupt is
periodically generated. Instead, an interval timer (i.e., one-shot timer/clock)
is used to program the time interval after which a clock interrupt is
generated.
3 pts

When a kernel is used in an environment where events are sparse (not densely
spaced in time) then using a tickful kernel may waste battery power in mobile
devices (or generate heat in wired devices) by executing lower half code
to handle system timer interrupts that are not necessary. A tickless kernel
that generates clock interrupts only when needed conserves energy (and
generates less heat).
3 pts

An interval timer is a hardware clock used to program when the next clock
interrupt should be generated.
2 pts

P3 20 pts

A page fault is an interrupt that is generated when a memory reference to
a page is made when executing the instructions of a process where the page
is not resident in main memory (i.e., stored on disk).
5 pts

Disk I/O (whether hard disk or SSD) is several orders of magnitude slower
than main memory access. Hardware busy waiting on disk I/O to complete would
waste CPU cycles.
// Instead, the process that page faulted is context-switched out and blocks
// until disk I/O completes to load the missing page in main memory. The CPU
// can be utilized by a ready process that is context-switched in. Performing
// scheduling and context-switch are the responsibility of kernels.
5 pts

Kernel performs disk read operation to read missing page into free frame in
memory. When disk read by lower half completes, the page faulting process is
unblocked and made ready.
// Scheduler than eventually selects the page faulting process to resume
// running by re-executing the instruction that caused the page fault.

4 pts

In general, caches need to be flushed during context-switch if they contain process specific content (data, instruction, page table entries). This causes cache misses when the context-switched in process executes.
3 pts

When processes execute in kernel mode due to system calls or handling interrupts, since upper and lower half kernel code reside at the same address in the virtual memory of all processes these entries in caches do not need to be flushed/invalidated during context-switch.
3 pts

Bonus 10 pts

Produce/consumer kernel buffer that sits between upper and lower half during video streaming. If the speed or rate of reading the buffer is lower than the write speed, buffer will become full and subsequent writes discarded. This results in loss of performance due to damage to video quality.
5 pts

Memory thrashing. If memory demand significantly exceeds available physical memory, page faults may evict pages to disk that are then needed in the near future. There is limit to locality of reference.
5 pts

// If other performance issues that are meaningful are noted, please assign
// partial credit.