CS354 Final Solution, fall 2023

P1(a) 20 pts

Each indexing block contains a number of direct indices/pointers (e.g., 10)
to data blocks which allow constant overhead indexing of small files.
4 pts

Each indexing blocks contains indirect, doubly indirect, triply indirect
pointers to indirect blocks that allow logarithmic overhead for large files.
4 pts

A direct index/pointer is generalized so that it specifies a range such that
if a large file is stored in contiguous data blocks its data block indices can
be represented by the range.
4 pts

FAT file systems are viewed as meeting the needs of mobile storage devices
such as USB thumb drives where a file resides in contiguous data blocks that
are not frequently updated, hence likely remain contiguous.
4 pts

Major role: wear leveling. It is important to prevent an erase block from
becoming prematurely defunct due to excessive number of write operations that
decrease its lifespan.
4 pts
// Garbage collection is fine too. With explanation of why essential gets 4 pts.

P1(b) 20 pts

Pro: fast/efficient.
Con: disruptive since important events may be ignored.
4 pts

The P() and V() system calls (wait() and signal() in XINU) themselves need to be
executed atomically which is achieved by using hardware support such as
interrupt disabling.
4 pts

Two counting semaphores: csem is initialized to 0, represents number of data items
in buffer; psem is initialized to buffer size, represents free buffer space.

Writer/producer: // Can be coded as infinite while-loop but unnecessary.
wait(psem); write to buffer; signal(csem);
4 pts

Reader/consumer: // Can be coded as infinite while-loop but unnecessary.
wait(csem); read from buffer; signal(psem);
4 pts

Producer executes first and writes a number of data items into buffer. Buffer
is large enough so that there is plenty of space. Producer is context-switched
out, consumer executes. With two counting semaphores consumer can proceed and
read from buffer since the part of the buffer that it reads from does not
overlap the part that the writer has been writing to. With a single semaphore the
entire buffer is locked (mutual exclusion), hence the consumer is blocked from
reading until the writer releases the semaphore. This incurs context-switching
overhead.
4 pts

P2(a) 20 pts

Assume fixed block size of K bytes. If data needed is S < K then K − S is wasted
space, called internal fragmentation.
2 pts

A system has total of M bytes of free memory but maximum contiguous memory is only
C < M bytes. A process requests S bytes where C < S < M, hence system call fails
even though there is sufficient total free memory. External fragmentation is
fragmenting of M total free space into smaller contiguous blocks (of size C or less).
3 pts

2^{20} which is around 1 million since 12 bits are consumed by offset within a 4 KB page.
2 pts

Contains 20-bit frame number that a 20-bit page number is translated to.
2 pts

In demand paging, due to memory pressure pages may be kept on disk and brought into main memory when referenced by processes.
2 pts

12 page offset bits are unused which are utilized as flag for purposes such as specifying whether a page is resident in main memory, whether it has been accessed (read/written), written (dirty bit) with hardware support.
3 pts

In a major page fault, a process accesses a virtual address whose page is not resident in main memory, i.e., has been evicted to disk. This generates a (major) page fault interrupt.
2 pts

A process can only access virtual addresses which are translated by per-process page tables into physical addresses. Since a kernel controls how page tables are configured, by translating virtual addresses of two processes to disjoint physical addresses it becomes impossible for a process to access the memory content of another process.
2 pts

Main drawback: context-switching overhead.
2 pts

P2(b) 20 pts

Writer: the current process whose context is being borrowed by the (bottom half of the) lower half in response to a device interrupt.
6 pts

Reader: the process that made the read system call which runs upper half kernel code to read from the kernel buffer.
6 pts

Since there is no space (and assuming the default drop tail FIFO discipline) the data is discarded.
4 pts

The reader is not able to match the pace of the writer. This is likely due to the scheduler not scheduling the process that makes read system calls frequently enough.
4 pts

P3 20 pts

Select page to evict that will not be accessed for the longest time in the future.
4 pts

It is optimal in the sense of minimizing the number of page faults.
2 pts

LRU is a good approximation because (a) it picks a page to evict from the past that has not been accessed for the longest time, (b) by locality of reference pages accessed in the recent past are likely to be accessed again in the near future (those far in the past not).
4 pts

Global clock periodically sweeps across all pages, deprecating the two bits xy 11 -> 10 -> 00 where x is set to 1 by hardware if a page has been accessed (i.e., read or written) and y is set to 1 if a page has been written to. Pages with bits 00 are targeted for eviction.
4 pts

The time needed to perform a complete sweep to return to a previously visited page

represents a time interval of "recency." If during this time interval a page is written to, its bits are set to 11 by hardware. If only read then 10. The time interval codifies "recently used" which is an approximation of "least recently used."
3 pts

A kernel process/thread is commonly used to carry out global clock.
3 pts

Bonus 10 pts

In memory thrashing recently evicted pages are needed again in the near future resulting in the system expending overhead to shuffle pages between main memory and disk which are significantly slower.
4 pts

High page fault rate, low CPU utilization, high disk I/O.
3 pts

Since CPU is underutilized due to waiting on slow disk I/O to complete, increasing CPU speed will have marginal/no benefit.
3 pts