CS354 Midterm Solution, spring 2020

P1(a) 20 pts

Upper half: respond to system calls.
4 pts

Lower half: respond to interrupts.
4 pts

The scheduler lies in-between the two halves.
3 pts

Upper half: resume() -> ready() -> resched()
3 pts

Lower half: clkdisp() -> clkhandler() -> resched()
3 pts

null process runs if there are no other ready processes.
3 pts

P1(b) 20 pts

XINU system calls are regular C function calls. They do not trap
from user mode to kernel mode.
4 pts

system call wrapper function, system call dispatcher, kernel code implementing system call.
3 pts

wrapper function: execute int instruction and pass arguments to dispatcher.
3 pts

dispatcher: based on EAX value, jump to getpid() or getprio(); in the latter,
pass PID argument.
3 pts

kernel code getpid(), getprio(): execute kernel function implementing system call.
3 pts

Changing from user mode to kernel mode.
Switching stacks.
4 pts

P1(c) 20 pts

Dequeue: constant overhead since pointer to first element suffices for extraction
of highest priority ready process.
3 pts

Eequeue: linear overhead since in the worst-case all processes in the ready list
must be compared.
3 pts

Solaris dequeue: constant overhead since in the worst-case loop over 60 priority
levels from 59 down to 0. At first non-empty priority level, dequeue first process
of the FIFO list at that priority level.
3 pts

Solaris enqueue: constant overhead since priority value serves as index into array
and inserting a process at the end of the FIFO list at that priority level can be done
in constant overhead with an end-of-list pointer.
3 pts

If a process consumes all of its time slice, treat it as CPU-bound and reduce priority.
If a process blocks without consuming all of its time slice, treat it as I/O-bound and
increase priority.
4 pts

If a process has spent more than 1 second (value is configurable) in ready state, increase
its priority so that likelihood of not getting CPU cycles for a prolonged period
(i.e., starvation) is reduced.

4 pts

P2(a) 20 pts

Pushing sequence:
function pointer (first argument of create()) as return address
EBP
EFLAGS
8 general-purpose registers
8 pts

IF flag is set to 1 so that when ctxsw() jumps/returns to the function pointer, the app code executes with interrupts enabled.
4 pts

Address of userret() (i.e., INITRET) is pushed on the stack before function pointer specified in the first argument of create() is pushed.
4 pts

userret() calls kills(getpid()) which frees up resources held by the process to be terminated and calls resched() to context-switch in another process.
4 pts

P2(b) 20 pts

Kernel actions:
Modify kernel stack of sleeping process so that upon switching from kernel mode to user mode after completion of sleep system call, the return address points to the callback function.
Modify user stack of sleeping process so that when callback function returns it jumps to the original return address at the time of sleep system call trap.
12 pts

Unless the priority of the sleeping process is high, after waking up and becoming ready, there may be a delay, small or large, before the process becomes current. In general, since sleeping processes are treated as I/O-bound and receive higher priority relative to CPU-bound processes, it is likely that the sleeping process after becoming ready will not have to incur significant wait time before becoming current. However, wait time depends on the priority of other ready processes and, in the worst-case, may be lengthy.
8 pts

Bonus 10 pts

An operating system is a collection (i.e., library) of functions that are invoked upon system call or interrupt in kernel mode by the current process.
4 pts

The upper half is the set of kernel functions invoked upon execution of system calls.
The lower half is the set of kernel functions invoked upon interrupt.
The null/idle process ensures that there is a process whose context can be borrowed to run lower half kernel code upon interrupt.
6 pts