Part-1

Question: What can be a problem with supporting such a system call in any OS? How could we mitigate this problem?

Answer: This system call allows the user to change the priority of a process. But actually this should be done in kernel mode only.

I have implemented 2 system calls : setPriority and printPriority.

setPriority set's the user given priority to the current running process.
While calling setPriority syscall in shell, one more argument should be given which is the value of priority.
It also gives an error if the user given priority is less than 0 or more than 9.
If priority is not given by the user, then it gives an error again.
If the syscall is successful, then it prints the information of all processes before changing the priority and after changing the priority.

printPriority is a system call which on calling in the shell prints the process's information along with its priority.
It can be used for checking the current status of processes.

Also I have initialised the default priority of all processes to 5 in allocproc() function in proc.c.

Part-2

Question: What is the default implementation policy in xv6?

Answer: Default uses round robin in process scheduling.

Question: There is a problem with such a priority based policy. What is that problem?
How can it be
mitigated?

Answer: In round-robin scheduling, starvation is possible. A process with less priority can suffer starvation. This can be mitigated using bounded waiting policy maybe so that a process with low priority may not have to starve.

Here, I have added few lines of code in scheduler() function in proc.c. There’s a for loop, which goes through all the processes in the page table and finds the process with highest priority and runs that process.