Kalena Galarnyk

CS354 Process and Priority Scheduling.

1. When running the above, what outputs do you observe? Are the 'P' characters printed by the parent process interleaved by outputs from its children? Explain your findings by tying the observed behavior to relevant XINU kernel code in system/ and the fact that XINU implements static priority scheduling.

   OUTPUT:

   P
   P
   P
   AAAAAAAAAABBBBBBBBBBCCCCCCCCCC

   When I ran my program as indicated, the 'P' characters are printed all before the outputs given by the children. This results from the main process having a higher priority than the children. The current XINU scheduling algorithm implements a round-robin algorithm for processes with equal priorities. However, as the main process has the highest priority, it is allowed to get the CPU time before the children.

2. Set the kernel parameter INITPRIO to 5 and rerun the experiment. Does the system behavior as gleaned from the generated output differ from Problem 4.1?

   OUTPUT:

   P
   AAAAAAAAAAP
   BBBBBBBBBBP
   CCCCCCCCCC

   In this case, while the main process is run, main prints 'P' before it calls resume on process A but then resumes process A (priority 10) while main process is using INITPRIO (5 rather than 20 this time). Process A is allowed to continue due to highest priority and uses CPU time until completed. When process A completes, the CPU returns its attention to the main process and prints 'P'. When main resumes process B, also with higher priority than main, CPU allows process B to complete before returning to main. The same happens when main resumes process C after printing the final 'P'.

3. Reset INITPRIO to the default value in Problem 4.1 but set the priority values of the three processes using create() to 6, 8, and 10 (for 'A', 'B', and 'C', respectively). How do the results compare to the output of Problem 4.1? What happens if INITPRIO is set to 6?

OUTPUT:
P
P
P
CCCCCCCCCCBBBBBBBBBBAAAAAAAAAA

While the three ‘P’s are printed first still, the process with the highest priority is queued first. Thus C is completed before B, and B is completed before A.

OUTPUT when if INITPRIO is set to 6:
P
P
P
CCCCCCCCCCBBBBBBBBBBAAAAAAAAAA

The output still seems the same, but only in part because C is the last one resumed. If C were the first process resumed, one P would print and then C would share CPU time with main. If C did not take too much time, main might get to print another P but C would complete before A and B.

4. What happens in Problem 4.1 if the priority of the null process is set to INITPRIO? What happens if it is set to 30?

   OUTPUT when null process is set to INITPRIO:

   P
   P
   P

   Because main and null process have the same priority, they will execute before process A, B, and C. Because all process with higher priority execute before lower priority, the null process consumes the CPU and A,B, and C never complete.

   OUTPUT when null process is set to 30:

   (NOTHING)

   The reason nothing is printed is that the null process is given priority higher than main. Therefore the CPU runs null process, because it is always running something and thus the reason to have a null process in the first place.

5. Modify printloop() such that its inner loop is replaced by a call to sleepms(). Also modify the XINU null process so that it outputs 'N' in its infinite loop (at the end of nulluser()). Rerun the experiment of Problem 4.1 with sleepms() argument set to 1 msec. What output do you observe? What happens if the argument to sleepms() is set to 0?

   OUTPUT when sleepms set to 1:
   P
   P
   P
   BBCCABBCCAABCCAABBCCAABBCC

   Main process recreating shell

   AB

   Main process recreating shell

   AA

   Main process recreating shell

   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN……..

   Basically the main process prints P but A, B, and C do not complete before main takes CPU back or another process takes.

   OUTPUT when sleepms set to 0:

   P
   P
   P
   ABBCCAABBCCAABCCAABBCCAABBCCABNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN……

   A,B, and C take turns with CPU time and then null process. Main does not recreate shell….