University of Windsor

School of Computer Science
Faculty of Science
COMP-2560: System Programming, Fall 2022

Lec#	Date	Title	Due Date	Grade Release Date
Lec09	Week 09	**Multiprocessing**	Two-Week Lec Nov. 23, 2022, Wednesday Midnigth	Nov. 28, 2022

The objectives of the weekly lecture assignments (Lecs) are to practice on topics covered in the lectures as well as improve the student's *critical thinking and problem-solving skills in ad hoc topics that are closely related but not covered in the lectures*. Lecture assignments also help students with research skills, including accessing, retrieving, and evaluating information (information literacy).

Lecture Assignments Deliverables

You should answer **two questions** below using an editor like MS Word, Notepad, and the likes or pen in papers. In the latter case, you must scan the papers clearly and merge them into a **single file** lec02_uwindid.pdf containing your name, uwinid, student#. **Please note that if your answers cannot be read, you will lose marks.** Please follow the naming convention as you lose marks otherwise. Instead of uwindid, use your own account name, e.g., mine is hfani@uwindsor.ca, so my submission would be: lec09_hfani.pdf

Lecture Assignments

Select two questions based on your preference!

1. Kernel shares a processor to multiple processes based on their *niceness*. Initially, a process is given a *nice* number NZERO which is a non-zero number. A process could choose to run with lower priority by increasing its nice value (being more generous) and reducing its share of the processor. To raise its priority and have more share of processor, the process should decrease its niceness (being meaner). How a process can do the following:
    a. Know the current nice value
    b. Increase the nice value (more generous)
    c. Decrease the nice value (meaner)
2. Do you think there should be a relationship between the size of a program and its nice value (priority)? Explain your answer.
3. Processes can be categorized into two groups: a) I/O-bound, b) CPU-bound. Explain each group and their differences.
4. In the multi-process environment, there is a chance for a process to starve! Explain the starvation problem and when it happens.
5. What happens in the following scenarios?
    a. The parent did wait(), the child has not reached its exit() and is suddenly killed by a user or the kernel?
    b. The parent forgets to call wait() and enters a busy waiting loop
6. Which one is more dangerous, orphans or zombies? Why?
7. Why is there no parent-child hierarchy when a process calls exec()?
8. At fork(), the parent has no opportunity to pass additional information to the newborn child process. Can you provide a workaround?
9. The memory layout of the child is a copy of its parent. So, any change to the variables cannot be seen by the parent. Can you provide a workaround that the child process can send information to its parent?
10. How can a process have a grandchild? Explain by an example.
11. How to retrieve the list of the following type of processes:
    a. Orphans
    b. Zombies
    c. Blocked
    d. Ready
    e. Running