COMP3500 – Frequently Asked Questions

Project 5 – CPU Scheduling

1. **CPU Usage.** What is the definition of CPU usage?

   Answer: In this project, CPU usage is an overall system performance metric measured as

   ```
    cpu_usage = total_burst_time / total_simulation_time;  (1)
   ```

   Please make use of equation (1) to compute CPU usage. For example, if it takes 20 time units to finish all the tasks and the total `burst_time` is 18, then the cpu_usage is 18/20 = 90%. Generally speaking, if process i has cpu_time_i and I/O_time_i, the CPU usage from the perspective of this process should be:

   ```
   cpu_usage_i = cpu_time_i / (cpu_time_i + I/O_time_i);   (2)
   ```

2. **Array Size.** C does not allow use of dynamic arrays. Since we don't know the length of the input text, should we use a static array with a large initial value (like size 100) or should we attempt to use malloc to simulate dynamic arrays?

   Answer: If you are comfortable of using dynamic data structure, you should create a singly linked lists to manage all the tasks. In case you store task information in a static array, you can define the maximal size of task_array as a large value (e.g., 64 or 128). This parameter can be easily configured as a system parameter in form of a constant.

3. **Segmentation fault (core dumped).** When implementing a loop to read the file in line by line I used

   ```
           while(fgets(line, sizeof line, ptr) != NULL)
   ```

   I keep receiving a segmentation fault error when reaching this line. When debugging in gdb I receive

   ```
   Program received signal SIGSEGV, Segmentation fault.
   _IO_fgets (buf=0x7fffffffe220 "", n=50, fp=0x0) at
   iofgets.c:47
   47        _IO_acquire_lock (fp);
   ```

   Is this due to the way I implemented fgets()?

   Answer: In my sample code (i.e., `input.c`) posted on Canvas, I leveraged the `fscanf()` function to read task information from an input file (i.e., fp – a file descriptor) into a static array of tasks. A portion of the sample code is given below:

```
/* open file */
file_name = argv[1];
if (! (fp = fopen(file_name, "r"))) {
      printf("File %s can't be opened.\n");
      return EXIT_FAILURE;
}
/*
  * Read data from input file.
 * Caveat: If you don’t place & before task_array[count],
 * you will receive a “core dumped” message when you run
 * your simulator.
 */
count = 0;
while (fscanf(fp, "%u %u %u", &task_array[count].pid, \
       &task_array[count].arrival_time,              \
                &task_array[count].burst_time)!= EOF) {
      count++;
}
```