CSCI 4210 — Operating Systems
Simulation Project Part II (document version 1.0)
Processes and CPU Scheduling

Overview

- This assignment is due in Submitty by 11:59PM EST on Thursday, August 15, 2024
- This project is to be completed either individually or in a team of at most three students; as with Project Part I, form your team within the Submitty gradeable, but do **not** submit any code until we announce that auto-grading is available
- **NEW:** If you worked on a team for Part I, feel free to change your team for Part II; all code is reusable from Part I even if you change teams
- Beyond your team (or yourself if working alone), **do not share your code**; however, feel free to discuss the project content and your findings with one another on our Discussion Forum
- To appease Submitty, you must use one of the following programming languages: C, C++, or Python (be sure you choose only **one** language for your entire implementation)
- You will have **five** penalty-free submissions on Submitty, after which points will slowly be deducted, e.g., `-1` on submission #6, etc.
- You can use at most three late days on this assignment; in such cases, each team member must use a late day
- You will have at least **three** days before the due date to submit your code to Submitty; if the auto-grading is not available three days before the due date, the due date will be 11:59PM EDT three days after auto-grading becomes available
- **NEW:** Given that your simulation results might not entirely match the expected output on Submitty, we will cap your auto-graded grade at **50 points** even though there will be more than 50 auto-graded points per language available in Submitty
- All submitted code **must** successfully compile and run on Submitty, which currently uses Ubuntu v22.04.4 LTS
- If you use C or C++, your program **must** successfully compile via `gcc` or `g++` with no warning messages when the `-Wall` (i.e., warn all) compiler option is used; we will also use `-Werror`, which will treat all warnings as critical errors; the `-lm` flag will also be included; the `gcc`/`g++` compiler is currently version 11.4.0 (`Ubuntu 11.4.0-1ubuntu1~22.04`)
- For source file naming conventions, be sure to use `*.c` for C and `*.cpp` for C++; in either case, you can also include `*.h` files
- For Python, you must use `python3`, which is currently Python 3.10.12; be sure to name your main Python file `project.py`; also be sure no warning messages or extraneous output occur during interpretation
- Please "flatten" all directory structures to a single directory of source files
- Note that you can use square brackets in your code

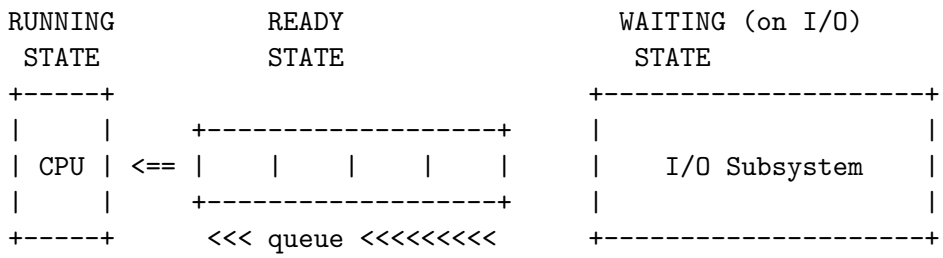
Project specifications

For Part II of our simulation project, given the set of processes pseudo-randomly generated in Part I, you will implement a series of simulations of a running operating system. The overall focus will again be on processes, assumed to be resident in memory, waiting to use the CPU. Memory and the I/O subsystem will not be covered in depth in either part of this project.

Conceptual design — (from Part I)

A **process** is defined as a program in execution. For this assignment, processes are in one of the following three states, corresponding to the picture shown further below.

- **RUNNING:** actively using the CPU and executing instructions
- **READY:** ready to use the CPU, i.e., ready to execute a CPU burst
- **WAITING:** blocked on I/O or some other event



Processes in the **READY** state reside in a queue called the ready queue. This queue is ordered based on a configurable CPU scheduling algorithm. You will implement specific CPU scheduling algorithms in Part II of this project.

All implemented algorithms (in Part II) will be simulated for the **same set of processes**, which will therefore support a comparative analysis of results. In Part I, the focus is on generating useful sets of processes via pseudo-random number generators.

Back to the conceptual model, when a process is in the **READY** state and reaches the front of the queue, once the CPU is free to accept the next process, the given process enters the **RUNNING** state and starts executing its CPU burst.

After each CPU burst is completed, if the process does not terminate, the process enters the **WAITING** state, waiting for an I/O operation to complete (e.g., waiting for data to be read in from a file). When the I/O operation completes, depending on the scheduling algorithm, the process either (1) returns to the **READY** state and is added to the ready queue or (2) preempts the currently running process and switches into the **RUNNING** state.

Note that preemptions occur only for certain algorithms.

Algorithms — (Part II)

The **four** algorithms that you must simulate are first-come-first-served (FCFS); shortest job first (SJF); shortest remaining time (SRT); and round robin (RR). When you run your program, all four algorithms are to be simulated in succession with the same initial set of processes.

Each algorithm is summarized below.

First-come-first-served (FCFS)

The FCFS algorithm is a non-preemptive algorithm in which processes simply line up in the ready queue, waiting to use the CPU. This is your baseline algorithm.

Shortest job first (SJF)

In SJF, processes are stored in the ready queue in order of priority based on their anticipated CPU burst times. More specifically, the process with the shortest predicted CPU burst time will be selected as the next process executed by the CPU. SJF is non-preemptive.

Shortest remaining time (SRT)

The SRT algorithm is a preemptive version of the SJF algorithm. In SRT, when a process arrives, if it has a predicted CPU burst time that is less than the remaining predicted time of the currently running process, a preemption occurs. When such a preemption occurs, the currently running process is added to the ready queue based on priority, i.e., based on its remaining predicted CPU burst time.

Round robin (RR)

The RR algorithm is essentially the FCFS algorithm with time slice t_{slice}. Each process is given t_{slice} amount of time to complete its CPU burst. If the time slice expires, the process is preempted and added to the end of the ready queue.

If a process completes its CPU burst before a time slice expiration, the next process on the ready queue is context-switched in to use the CPU.

For your simulation, if a preemption occurs and there are no other processes on the ready queue, do **not** perform a context switch. For example, given process `G` is using the CPU and the ready queue is empty, if process `G` is preempted by a time slice expiration, do not context-switch process `G` back to the empty queue; instead, keep process `G` running with the CPU and do **not** count this as a context switch. In other words, when the time slice expires, check the queue to determine if a context switch should occur.

Simulation configuration — (extended from Part I)

The key to designing a useful simulation is to provide a number of configurable parameters. This allows you to simulate and tune for a variety of scenarios, e.g., a large number of CPU-bound processes, differing average process interarrival times, multiple CPUs, etc.

Define the simulation parameters shown below as tunable constants within your code, all of which will be given as command-line arguments. In Part II of the project, additional parameters will be added.

- `*(argv+1)`: Define n as the number of processes to simulate. Process IDs are assigned a two-character code consisting of an uppercase letter from `A` to `Z` followed by a number from `0` to `9`. Processes are assigned in order `A0`, `A1`, `A2`, ..., `A9`, `B0`, `B1`, ..., `Z9`.

- `*(argv+2)`: Define n_{cpu} as the number of processes that are CPU-bound. For this project, we will classify processes as I/O-bound or CPU-bound. The n_{cpu} CPU-bound processes, when generated, will have CPU burst times that are longer by a factor of 4 and will have I/O burst times that are shorter by a factor of 8.

- `*(argv+3)`: We will use a pseudo-random number generator to determine the *interarrival times* of CPU bursts. This command-line argument, i.e. *seed*, serves as the seed for the pseudo-random number sequence. To ensure predictability and repeatability, use `srand48()` with this given seed before simulating **each** scheduling algorithm and `drand48()` to obtain the next value in the range $[0.0, 1.0)$. Since Python does not have these functions, implement an equivalent 48-bit linear congruential generator, as described in the `man` page for these functions in C.[1]

- `*(argv+4)`: To determine interarrival times, we will use an exponential distribution, as illustrated in the `exp-random.c` example. This command-line argument is parameter λ; remember that $\frac{1}{\lambda}$ will be the average random value generated, e.g., if $\lambda = 0.01$, then the average should be appoximately 100.

  In the `exp-random.c` example, use the formula shown in the code, i.e., $\dfrac{-\ln r}{\lambda}$.

- `*(argv+5)`: For the exponential distribution, this command-line argument represents the upper bound for valid pseudo-random numbers. This threshold is used to avoid values far down the long tail of the exponential distribution. As an example, if this is set to 3000, all generated values above 3000 should be skipped. For cases in which this value is used in the ceiling function (see the next page), be sure the ceiling is still valid according to this upper bound.

- `*(argv+6)`: Define t_{cs} as the time, in milliseconds, that it takes to perform a context switch. Specifically, the first half of the context switch time (i.e., $\frac{t_{cs}}{2}$) is the time required to remove the given process from the CPU; the second half of the context switch time is the time required to bring the next process in to use the CPU. Therefore, require t_{cs} to be a positive even integer.

[1] Feel free to post your code for this on the Discussion Forum for others to use.

- `*(argv+7)`: For the SJF and SRT algorithms, since we do not know the actual CPU burst times beforehand, we will rely on estimates determined via exponential averaging. As such, this command-line argument is the constant α, which must be a numeric floating-point value in the range $[0, 1]$.

  Note that the initial guess for each process is $\tau_0 = \dfrac{1}{\lambda}$.

  Also, when calculating τ values, use the "ceiling" function for all calculations.

- `*(argv+8)`: For the RR algorithm, define the time slice value, t_{slice}, measured in milliseconds. Require t_{slice} to be a positive integer.

Pseudo-random numbers and predictability — (from Part I)

A key aspect of this assignment is to compare the results of each of the simulated algorithms with one another given the same initial conditions, i.e., the same initial set of processes.

To ensure each CPU scheduling algorithm runs with the same set of processes, carefully follow the algorithm below to create the set of processes.

For each of the n processes, in order `A0` through `Z9`, perform the steps below, with CPU-bound processes generated first. Note that all generated values are integers.

Define your exponential distribution pseudo-random number generation function as `next_exp()` (or another similar name).

1. Identify the initial process arrival time as the "floor" of the next random number in the sequence given by `next_exp()`; note that you could therefore have a zero arrival time
2. Identify the number of CPU bursts for the given process as the "ceiling" of the next random number generated from the **uniform distribution** obtained via `drand48()` multiplied by 32; this should obtain a random integer in the inclusive range $[1, 32]$
3. For *each* of these CPU bursts, identify the CPU burst time and the I/O burst time as the "ceiling" of the next two random numbers in the sequence given by `next_exp()`; multiply the I/O burst time by 8 such that I/O burst time is close to an order of magnitude longer than CPU burst time; as noted above, for CPU-bound processes, multiply the CPU burst time by 4 and divide the I/O burst time by 8 (i.e., do not bother multiplying the original I/O burst time by 8 in this case); for the last CPU burst, do not generate an I/O burst time (since each process ends with a final CPU burst)

Simulation specifics — (Part II)

Your simulator keeps track of elapsed time t (measured in milliseconds), which is initially zero for each scheduling algorithm. As your simulation proceeds, t advances to each "interesting" event that occurs, displaying a specific line of output that describes each event.

The "interesting" events are:

- Start of simulation for a specific algorithm
- Process arrival (i.e., initially and at each I/O completion)
- Process starts using the CPU
- Process finishes using the CPU (i.e., completes a CPU burst)
- Process has its τ value recalculated (i.e., after a CPU burst completion)
- Process preemption (SRT and RR only)
- Process starts an I/O burst
- Process finishes an I/O burst
- Process terminates by finishing its last CPU burst
- End of simulation for a specific algorithm

Note that the "process arrival" event occurs each time a process arrives, which includes both the initial arrival time and when a process completes an I/O burst. In other words, processes "arrive" within the subsystem that consists only of the CPU and the ready queue.

The "process preemption" event occurs each time a process is preempted. When a preemption occurs, a context switch occurs, except when the ready queue is empty for the RR algorithm.

After you simulate each scheduling algorithm, you must reset your simulation back to the initial set of processes and set your elapsed time back to zero.

Note that there may be times during your simulation in which the simulated CPU is idle because no processes have arrived yet or all processes are busy performing I/O. Also, your simulation ends when all processes terminate.

If different types of events occur at the same time, simulate these events in the following order: (a) CPU burst completion; (b) process starts using the CPU; (c) I/O burst completions; and (d) new process arrivals.

Further, any "ties" that occur *within* one of these categories are to be broken using process ID order. As an example, if processes `G1` and `S9` happen to both complete I/O bursts at the same time, process `G1` wins this "tie" (because `G1` is lexicographically before `S9`) and is therefore added to the ready queue before process `S9`.

Be sure you do not implement any additional logic for the I/O subsystem. In other words, there are no specific I/O queues to implement.

Measurements — (from Part I)

There are a number of measurements you will want to track in your simulation. For each algorithm, you will count the number of preemptions and the number of context switches that occur. Further, you will measure CPU utilization by tracking CPU usage and CPU idle time.

Specifically, for **each CPU burst**, you will track CPU burst time (given), turnaround time, and wait time.

CPU burst time

CPU burst times are randomly generated for each process that you simulate via the above algorithm. CPU burst time is defined as the amount of time a process is **actually** using the CPU. Therefore, this measure does not include context switch times.

Turnaround time

Turnaround times are to be measured for each process that you simulate. Turnaround time is defined as the end-to-end time a process spends in executing a **single CPU burst**.

More specifically, this is measured from process arrival time through to when the CPU burst is completed and the process is switched out of the CPU. Therefore, this measure includes the second half of the initial context switch in and the first half of the final context switch out, as well as any other context switches that occur while the CPU burst is being completed (i.e., due to preemptions).

Wait time

Wait times are to be measured **for each CPU burst**. Wait time is defined as the amount of time a process spends waiting to use the CPU, which equates to the amount of time the given process is actually in the ready queue. Therefore, this measure does not include context switch times that the given process experiences, i.e., only measure the time the given process is actually in the ready queue.

CPU utilization

Calculate CPU utilization by tracking how much time the CPU is actively running CPU bursts versus total elapsed simulation time.

Required terminal output — (extended from Part I)

For Part II, required terminal output shows abbreviated Part I output followed by output for each of the four simulated algorithms.

Your simulator must display results for each of the four algorithms you simulate. For each algorithm, display a summary for each "interesting" event that occurs from time 0 through time 9999. For time 10000 and beyond, only display process termination events and the final end-of-simulation event. Example output files will be posted to Submitty.

Your simulator must display a line of output for each "interesting" event that occurs using the format shown below. Note that the contents of the ready queue are shown for each event.

```
time <t>ms: <event-details> [Q <queue-contents>]
```

The output format must follow the example shown below.

```
bash$ ./a.out 3 1 32 0.001 1024 4 0.75 256
<<< PROJECT PART I
<<< -- process set (n=3) with 1 CPU-bound process
<<< -- seed=32; lambda=0.001000; bound=1024
CPU-bound process A0: arrival time 319ms; 25 CPU bursts:
I/O-bound process A1: arrival time 506ms; 5 CPU bursts:
I/O-bound process A2: arrival time 821ms; 15 CPU bursts:

<<< PROJECT PART II
<<< -- t_cs=4ms; alpha=0.75; t_slice=256ms
time 0ms: Simulator started for FCFS [Q empty]
time 319ms: Process A0 arrived; added to ready queue [Q A0]
time 321ms: Process A0 started using the CPU for 1448ms burst [Q empty]
time 506ms: Process A1 arrived; added to ready queue [Q A1]
time 821ms: Process A2 arrived; added to ready queue [Q A1 A2]
time 1769ms: Process A0 completed a CPU burst; 24 bursts to go [Q A1 A2]
time 1769ms: Process A0 switching out of CPU; blocking on I/O until time 2379ms [Q A1 A2]
time 1773ms: Process A1 started using the CPU for 971ms burst [Q A2]
time 2379ms: Process A0 completed I/O; added to ready queue [Q A2 A0]
time 2744ms: Process A1 completed a CPU burst; 4 bursts to go [Q A2 A0]
time 2744ms: Process A1 switching out of CPU; blocking on I/O until time 3178ms [Q A2 A0]
time 2748ms: Process A2 started using the CPU for 408ms burst [Q A0]
time 3156ms: Process A2 completed a CPU burst; 14 bursts to go [Q A0]
time 3156ms: Process A2 switching out of CPU; blocking on I/O until time 8670ms [Q A0]
time 3160ms: Process A0 started using the CPU for 316ms burst [Q empty]
time 3178ms: Process A1 completed I/O; added to ready queue [Q A1]
time 3476ms: Process A0 completed a CPU burst; 23 bursts to go [Q A1]
time 3476ms: Process A0 switching out of CPU; blocking on I/O until time 3952ms [Q A1]
time 3480ms: Process A1 started using the CPU for 129ms burst [Q empty]
time 3609ms: Process A1 completed a CPU burst; 3 bursts to go [Q empty]
time 3609ms: Process A1 switching out of CPU; blocking on I/O until time 4875ms [Q empty]
```

```
time 3952ms: Process A0 completed I/O; added to ready queue [Q A0]
time 3954ms: Process A0 started using the CPU for 3556ms burst [Q empty]
time 4875ms: Process A1 completed I/O; added to ready queue [Q A1]
time 7510ms: Process A0 completed a CPU burst; 22 bursts to go [Q A1]
time 7510ms: Process A0 switching out of CPU; blocking on I/O until time 8476ms [Q A1]
time 7514ms: Process A1 started using the CPU for 188ms burst [Q empty]
time 7702ms: Process A1 completed a CPU burst; 2 bursts to go [Q empty]
time 7702ms: Process A1 switching out of CPU; blocking on I/O until time 9200ms [Q empty]
time 8476ms: Process A0 completed I/O; added to ready queue [Q A0]
time 8478ms: Process A0 started using the CPU for 2516ms burst [Q empty]
time 8670ms: Process A2 completed I/O; added to ready queue [Q A2]
time 9200ms: Process A1 completed I/O; added to ready queue [Q A2 A1]
time 12891ms: Process A1 terminated [Q empty]
time 52148ms: Process A0 terminated [Q empty]
time 66002ms: Process A2 terminated [Q empty]
time 66004ms: Simulator ended for FCFS [Q empty]

time 0ms: Simulator started for SJF [Q empty]
time 319ms: Process A0 (tau 1000ms) arrived; added to ready queue [Q A0]
time 321ms: Process A0 (tau 1000ms) started using the CPU for 1448ms burst [Q empty]
time 506ms: Process A1 (tau 1000ms) arrived; added to ready queue [Q A1]
time 821ms: Process A2 (tau 1000ms) arrived; added to ready queue [Q A1 A2]
time 1769ms: Process A0 (tau 1000ms) completed a CPU burst; 24 bursts to go [Q A1 A2]
time 1769ms: Recalculated tau for process A0: old tau 1000ms ==> new tau 1336ms [Q A1 A2]
time 1769ms: Process A0 switching out of CPU; blocking on I/O until time 2379ms [Q A1 A2]
time 1773ms: Process A1 (tau 1000ms) started using the CPU for 971ms burst [Q A2]
time 2379ms: Process A0 (tau 1336ms) completed I/O; added to ready queue [Q A2 A0]
time 2744ms: Process A1 (tau 1000ms) completed a CPU burst; 4 bursts to go [Q A2 A0]
time 2744ms: Recalculated tau for process A1: old tau 1000ms ==> new tau 979ms [Q A2 A0]
time 2744ms: Process A1 switching out of CPU; blocking on I/O until time 3178ms [Q A2 A0]
time 2748ms: Process A2 (tau 1000ms) started using the CPU for 408ms burst [Q A0]
time 3156ms: Process A2 (tau 1000ms) completed a CPU burst; 14 bursts to go [Q A0]
time 3156ms: Recalculated tau for process A2: old tau 1000ms ==> new tau 556ms [Q A0]
time 3156ms: Process A2 switching out of CPU; blocking on I/O until time 8670ms [Q A0]
time 3160ms: Process A0 (tau 1336ms) started using the CPU for 316ms burst [Q empty]
time 3178ms: Process A1 (tau 979ms) completed I/O; added to ready queue [Q A1]
time 3476ms: Process A0 (tau 1336ms) completed a CPU burst; 23 bursts to go [Q A1]
time 3476ms: Recalculated tau for process A0: old tau 1336ms ==> new tau 571ms [Q A1]
time 3476ms: Process A0 switching out of CPU; blocking on I/O until time 3952ms [Q A1]
time 3480ms: Process A1 (tau 979ms) started using the CPU for 129ms burst [Q empty]
time 3609ms: Process A1 (tau 979ms) completed a CPU burst; 3 bursts to go [Q empty]
time 3609ms: Recalculated tau for process A1: old tau 979ms ==> new tau 342ms [Q empty]
time 3609ms: Process A1 switching out of CPU; blocking on I/O until time 4875ms [Q empty]
time 3952ms: Process A0 (tau 571ms) completed I/O; added to ready queue [Q A0]
time 3954ms: Process A0 (tau 571ms) started using the CPU for 3556ms burst [Q empty]
time 4875ms: Process A1 (tau 342ms) completed I/O; added to ready queue [Q A1]
time 7510ms: Process A0 (tau 571ms) completed a CPU burst; 22 bursts to go [Q A1]
time 7510ms: Recalculated tau for process A0: old tau 571ms ==> new tau 2810ms [Q A1]
```

```
time 7510ms: Process A0 switching out of CPU; blocking on I/O until time 8476ms [Q A1]
time 7514ms: Process A1 (tau 342ms) started using the CPU for 188ms burst [Q empty]
time 7702ms: Process A1 (tau 342ms) completed a CPU burst; 2 bursts to go [Q empty]
time 7702ms: Recalculated tau for process A1: old tau 342ms ==> new tau 227ms [Q empty]
time 7702ms: Process A1 switching out of CPU; blocking on I/O until time 9200ms [Q empty]
time 8476ms: Process A0 (tau 2810ms) completed I/O; added to ready queue [Q A0]
time 8478ms: Process A0 (tau 2810ms) started using the CPU for 2516ms burst [Q empty]
time 8670ms: Process A2 (tau 556ms) completed I/O; added to ready queue [Q A2]
time 9200ms: Process A1 (tau 227ms) completed I/O; added to ready queue [Q A1 A2]
time 12705ms: Process A1 terminated [Q empty]
time 52148ms: Process A0 terminated [Q empty]
time 66002ms: Process A2 terminated [Q empty]
time 66004ms: Simulator ended for SJF [Q empty]

time 0ms: Simulator started for SRT [Q empty]
time 319ms: Process A0 (tau 1000ms) arrived; added to ready queue [Q A0]
time 321ms: Process A0 (tau 1000ms) started using the CPU for 1448ms burst [Q empty]
time 506ms: Process A1 (tau 1000ms) arrived; added to ready queue [Q A1]
time 821ms: Process A2 (tau 1000ms) arrived; added to ready queue [Q A1 A2]
time 1769ms: Process A0 (tau 1000ms) completed a CPU burst; 24 bursts to go [Q A1 A2]
time 1769ms: Recalculated tau for process A0: old tau 1000ms ==> new tau 1336ms [Q A1 A2]
time 1769ms: Process A0 switching out of CPU; blocking on I/O until time 2379ms [Q A1 A2]
time 1773ms: Process A1 (tau 1000ms) started using the CPU for 971ms burst [Q A2]
time 2379ms: Process A0 (tau 1336ms) completed I/O; added to ready queue [Q A2 A0]
time 2744ms: Process A1 (tau 1000ms) completed a CPU burst; 4 bursts to go [Q A2 A0]
time 2744ms: Recalculated tau for process A1: old tau 1000ms ==> new tau 979ms [Q A2 A0]
time 2744ms: Process A1 switching out of CPU; blocking on I/O until time 3178ms [Q A2 A0]
time 2748ms: Process A2 (tau 1000ms) started using the CPU for 408ms burst [Q A0]
time 3156ms: Process A2 (tau 1000ms) completed a CPU burst; 14 bursts to go [Q A0]
time 3156ms: Recalculated tau for process A2: old tau 1000ms ==> new tau 556ms [Q A0]
time 3156ms: Process A2 switching out of CPU; blocking on I/O until time 8670ms [Q A0]
time 3160ms: Process A0 (tau 1336ms) started using the CPU for 316ms burst [Q empty]
time 3178ms: Process A1 (tau 979ms) completed I/O; preempting A0 (predicted remaining time
  1318ms) [Q A1]
time 3182ms: Process A1 (tau 979ms) started using the CPU for 129ms burst [Q A0]
time 3311ms: Process A1 (tau 979ms) completed a CPU burst; 3 bursts to go [Q A0]
time 3311ms: Recalculated tau for process A1: old tau 979ms ==> new tau 342ms [Q A0]
time 3311ms: Process A1 switching out of CPU; blocking on I/O until time 4577ms [Q A0]
time 3315ms: Process A0 (tau 1336ms) started using the CPU for remaining 298ms of 316ms
  burst [Q empty]
time 3613ms: Process A0 (tau 1336ms) completed a CPU burst; 23 bursts to go [Q empty]
time 3613ms: Recalculated tau for process A0: old tau 1336ms ==> new tau 571ms [Q empty]
time 3613ms: Process A0 switching out of CPU; blocking on I/O until time 4089ms [Q empty]
time 4089ms: Process A0 (tau 571ms) completed I/O; added to ready queue [Q A0]
time 4091ms: Process A0 (tau 571ms) started using the CPU for 3556ms burst [Q empty]
time 4577ms: Process A1 (tau 342ms) completed I/O; added to ready queue [Q A1]
time 7647ms: Process A0 (tau 571ms) completed a CPU burst; 22 bursts to go [Q A1]
time 7647ms: Recalculated tau for process A0: old tau 571ms ==> new tau 2810ms [Q A1]
```

```
time 7647ms: Process A0 switching out of CPU; blocking on I/O until time 8613ms [Q A1]
time 7651ms: Process A1 (tau 342ms) started using the CPU for 188ms burst [Q empty]
time 7839ms: Process A1 (tau 342ms) completed a CPU burst; 2 bursts to go [Q empty]
time 7839ms: Recalculated tau for process A1: old tau 342ms ==> new tau 227ms [Q empty]
time 7839ms: Process A1 switching out of CPU; blocking on I/O until time 9337ms [Q empty]
time 8613ms: Process A0 (tau 2810ms) completed I/O; added to ready queue [Q A0]
time 8615ms: Process A0 (tau 2810ms) started using the CPU for 2516ms burst [Q empty]
time 8670ms: Process A2 (tau 556ms) completed I/O; preempting A0 (predicted remaining time
  2755ms) [Q A2]
time 8674ms: Process A2 (tau 556ms) started using the CPU for 182ms burst [Q A0]
time 8856ms: Process A2 (tau 556ms) completed a CPU burst; 13 bursts to go [Q A0]
time 8856ms: Recalculated tau for process A2: old tau 556ms ==> new tau 276ms [Q A0]
time 8856ms: Process A2 switching out of CPU; blocking on I/O until time 12602ms [Q A0]
time 8860ms: Process A0 (tau 2810ms) started using the CPU for remaining 2461ms of 2516ms
  burst [Q empty]
time 9337ms: Process A1 (tau 227ms) completed I/O; preempting A0 (predicted remaining time
  2278ms) [Q A1]
time 9341ms: Process A1 (tau 227ms) started using the CPU for 302ms burst [Q A0]
time 9643ms: Process A1 (tau 227ms) completed a CPU burst; 1 burst to go [Q A0]
time 9643ms: Recalculated tau for process A1: old tau 227ms ==> new tau 284ms [Q A0]
time 9643ms: Process A1 switching out of CPU; blocking on I/O until time 10973ms [Q A0]
time 9647ms: Process A0 (tau 2810ms) started using the CPU for remaining 1984ms of 2516ms
  burst [Q empty]
time 11050ms: Process A1 terminated [Q A0]
time 53446ms: Process A0 terminated [Q empty]
time 60792ms: Process A2 terminated [Q empty]
time 60794ms: Simulator ended for SRT [Q empty]

time 0ms: Simulator started for RR [Q empty]
time 319ms: Process A0 arrived; added to ready queue [Q A0]
time 321ms: Process A0 started using the CPU for 1448ms burst [Q empty]
time 506ms: Process A1 arrived; added to ready queue [Q A1]
time 577ms: Time slice expired; preempting process A0 with 1192ms remaining [Q A1]
time 581ms: Process A1 started using the CPU for 971ms burst [Q A0]
time 821ms: Process A2 arrived; added to ready queue [Q A0 A2]
time 837ms: Time slice expired; preempting process A1 with 715ms remaining [Q A0 A2]
time 841ms: Process A0 started using the CPU for remaining 1192ms of 1448ms burst [Q A2 A1]
time 1097ms: Time slice expired; preempting process A0 with 936ms remaining [Q A2 A1]
time 1101ms: Process A2 started using the CPU for 408ms burst [Q A1 A0]
time 1357ms: Time slice expired; preempting process A2 with 152ms remaining [Q A1 A0]
time 1361ms: Process A1 started using the CPU for remaining 715ms of 971ms burst [Q A0 A2]
time 1617ms: Time slice expired; preempting process A1 with 459ms remaining [Q A0 A2]
time 1621ms: Process A0 started using the CPU for remaining 936ms of 1448ms burst [Q A2 A1]
time 1877ms: Time slice expired; preempting process A0 with 680ms remaining [Q A2 A1]
time 1881ms: Process A2 started using the CPU for remaining 152ms of 408ms burst [Q A1 A0]
time 2033ms: Process A2 completed a CPU burst; 14 bursts to go [Q A1 A0]
time 2033ms: Process A2 switching out of CPU; blocking on I/O until time 7547ms [Q A1 A0]
time 2037ms: Process A1 started using the CPU for remaining 459ms of 971ms burst [Q A0]
```

```
time 2293ms: Time slice expired; preempting process A1 with 203ms remaining [Q A0]
time 2297ms: Process A0 started using the CPU for remaining 680ms of 1448ms burst [Q A1]
time 2553ms: Time slice expired; preempting process A0 with 424ms remaining [Q A1]
time 2557ms: Process A1 started using the CPU for remaining 203ms of 971ms burst [Q A0]
time 2760ms: Process A1 completed a CPU burst; 4 bursts to go [Q A0]
time 2760ms: Process A1 switching out of CPU; blocking on I/O until time 3194ms [Q A0]
time 2764ms: Process A0 started using the CPU for remaining 424ms of 1448ms burst [Q empty]
time 3020ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 3188ms: Process A0 completed a CPU burst; 24 bursts to go [Q empty]
time 3188ms: Process A0 switching out of CPU; blocking on I/O until time 3798ms [Q empty]
time 3194ms: Process A1 completed I/O; added to ready queue [Q A1]
time 3196ms: Process A1 started using the CPU for 129ms burst [Q empty]
time 3325ms: Process A1 completed a CPU burst; 3 bursts to go [Q empty]
time 3325ms: Process A1 switching out of CPU; blocking on I/O until time 4591ms [Q empty]
time 3798ms: Process A0 completed I/O; added to ready queue [Q A0]
time 3800ms: Process A0 started using the CPU for 316ms burst [Q empty]
time 4056ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 4116ms: Process A0 completed a CPU burst; 23 bursts to go [Q empty]
time 4116ms: Process A0 switching out of CPU; blocking on I/O until time 4592ms [Q empty]
time 4591ms: Process A1 completed I/O; added to ready queue [Q A1]
time 4592ms: Process A0 completed I/O; added to ready queue [Q A0]
time 4593ms: Process A1 started using the CPU for 188ms burst [Q A0]
time 4781ms: Process A1 completed a CPU burst; 2 bursts to go [Q A0]
time 4781ms: Process A1 switching out of CPU; blocking on I/O until time 6279ms [Q A0]
time 4785ms: Process A0 started using the CPU for 3556ms burst [Q empty]
time 5041ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 5297ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 5553ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 5809ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 6065ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 6279ms: Process A1 completed I/O; added to ready queue [Q A1]
time 6321ms: Time slice expired; preempting process A0 with 2020ms remaining [Q A1]
time 6325ms: Process A1 started using the CPU for 302ms burst [Q A0]
time 6581ms: Time slice expired; preempting process A1 with 46ms remaining [Q A0]
time 6585ms: Process A0 started using the CPU for remaining 2020ms of 3556ms burst [Q A1]
time 6841ms: Time slice expired; preempting process A0 with 1764ms remaining [Q A1]
time 6845ms: Process A1 started using the CPU for remaining 46ms of 302ms burst [Q A0]
time 6891ms: Process A1 completed a CPU burst; 1 burst to go [Q A0]
time 6891ms: Process A1 switching out of CPU; blocking on I/O until time 8221ms [Q A0]
time 6895ms: Process A0 started using the CPU for remaining 1764ms of 3556ms burst [Q empty]
time 7151ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 7407ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 7547ms: Process A2 completed I/O; added to ready queue [Q A2]
time 7663ms: Time slice expired; preempting process A0 with 996ms remaining [Q A2]
time 7667ms: Process A2 started using the CPU for 182ms burst [Q A0]
time 7849ms: Process A2 completed a CPU burst; 13 bursts to go [Q A0]
time 7849ms: Process A2 switching out of CPU; blocking on I/O until time 11595ms [Q A0]
time 7853ms: Process A0 started using the CPU for remaining 996ms of 3556ms burst [Q empty]
```

```
time 8109ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 8221ms: Process A1 completed I/O; added to ready queue [Q A1]
time 8365ms: Time slice expired; preempting process A0 with 484ms remaining [Q A1]
time 8369ms: Process A1 started using the CPU for 73ms burst [Q A0]
time 8442ms: Process A1 terminated [Q A0]
time 8446ms: Process A0 started using the CPU for remaining 484ms of 3556ms burst [Q empty]
time 8702ms: Time slice expired; no preemption because ready queue is empty [Q empty]
time 8930ms: Process A0 completed a CPU burst; 22 bursts to go [Q empty]
time 8930ms: Process A0 switching out of CPU; blocking on I/O until time 9896ms [Q empty]
time 9896ms: Process A0 completed I/O; added to ready queue [Q A0]
time 9898ms: Process A0 started using the CPU for 2516ms burst [Q empty]
time 54903ms: Process A0 terminated [Q empty]
time 61754ms: Process A2 terminated [Q empty]
time 61756ms: Simulator ended for RR [Q empty]
```

Required output file — (extended from Part I)

In addition to the above output (which should be sent to `stdout`), generate an output file called `simout.txt` that contains statistics for each simulated algorithm. For Part I, the file format is shown below (with `#` as a placeholder for actual numerical data). Use the "ceiling" function out to exactly three digits after the decimal point for your averages. If calculating an average will cause division by 0, simply set the average to 0.

```
-- number of processes: #
-- number of CPU-bound processes: #
-- number of I/O-bound processes: #
-- CPU-bound average CPU burst time: #.### ms
-- I/O-bound average CPU burst time: #.### ms
-- overall average CPU burst time: #.### ms
-- CPU-bound average I/O burst time: #.### ms
-- I/O-bound average I/O burst time: #.### ms
-- overall average I/O burst time: #.### ms
```

In addition to the Part I `simout.txt` output shown above, for Part II, append the output below for each simulated algorithm (with `#` as a placeholder for actual numerical data). As with Part I, use the "ceiling" function out to exactly three digits after the decimal point for your averages.

```
Algorithm <algorithm>
-- CPU utilization: #.###%
-- CPU-bound average wait time: #.### ms
-- I/O-bound average wait time: #.### ms
-- overall average wait time: #.### ms
-- CPU-bound average turnaround time: #.### ms
-- I/O-bound average turnaround time: #.### ms
-- overall average turnaround time: #.### ms
-- CPU-bound number of context switches: #
-- I/O-bound number of context switches: #
-- overall number of context switches: #
-- CPU-bound number of preemptions: #
-- I/O-bound number of preemptions: #
-- overall number of preemptions: #
```

For the RR algorithm, also include the following (appended to the end):

```
-- CPU-bound percentage of CPU bursts completed within one time slice: #.###%
-- I/O-bound percentage of CPU bursts completed within one time slice: #.###%
-- overall percentage of CPU bursts completed within one time slice: #.###%
```

Continuing the example from the previous page, the `simout.txt` file is as shown below. A set of full `simout.txt` output files will be posted on Submitty.

```
-- number of processes: 3
-- number of CPU-bound processes: 1
-- number of I/O-bound processes: 2
-- CPU-bound average CPU burst time: 1600.640 ms
-- I/O-bound average CPU burst time: 247.300 ms
-- overall average CPU burst time: 999.156 ms
-- CPU-bound average I/O burst time: 419.084 ms
-- I/O-bound average I/O burst time: 3277.778 ms
-- overall average I/O burst time: 1644.239 ms

Algorithm FCFS
-- CPU utilization: 68.121%
-- CPU-bound average wait time: 66.280 ms
-- I/O-bound average wait time: 677.200 ms
-- overall average wait time: 337.800 ms
-- CPU-bound average turnaround time: 1670.920 ms
-- I/O-bound average turnaround time: 928.500 ms
-- overall average turnaround time: 1340.956 ms
-- CPU-bound number of context switches: 25
-- I/O-bound number of context switches: 20
-- overall number of context switches: 45
-- CPU-bound number of preemptions: 0
-- I/O-bound number of preemptions: 0
-- overall number of preemptions: 0

Algorithm SJF
-- CPU utilization: 68.121%
-- CPU-bound average wait time: 66.280 ms
-- I/O-bound average wait time: 667.900 ms
-- overall average wait time: 333.667 ms
-- CPU-bound average turnaround time: 1670.920 ms
-- I/O-bound average turnaround time: 919.200 ms
-- overall average turnaround time: 1336.823 ms
-- CPU-bound number of context switches: 25
-- I/O-bound number of context switches: 20
-- overall number of context switches: 45
-- CPU-bound number of preemptions: 0
-- I/O-bound number of preemptions: 0
-- overall number of preemptions: 0

Algorithm SRT
...

Algorithm RR
...
```

Error handling — (from Part I)

If improper command-line arguments are given, report an error message to `stderr` and abort further program execution. Be sure that the number of processes is greater than zero; further, deduce other valid input value ranges from page 3.

In general, if an error is encountered, display a meaningful error message on `stderr`, then abort further program execution.

Error messages must be one line only and use the following format:

```
ERROR: <error-text-here>
```

Submission instructions — (from Part I)

To submit your assignment and also perform final testing of your code, please use Submitty.

Note that this assignment will be available on Submitty a minimum of three days before the due date. Please do not ask when Submitty will be available, as you should first perform adequate testing on your own Ubuntu platform.

Relinquishing allocated resources — (from Part I)

Be sure that all resources (e.g., dynamically allocated memory) are properly relinquished for whatever language/platform you use for this assignment. Sloppy programming will likely result in lower grades. Consider doing frequent code reviews with your teammates if working on a team.

Analysis questions to think about — (Part II)

As you work on this project, review your output and consider the questions below.

Though there is no formal write-up required for this project, you should be able to answer at least some of these questions based on specific results from your simulation. A few of these questions would require features beyond the given requirements.

1. Of the four simulated algorithms, which algorithm is the "best" algorithm for CPU-bound processes? Which algorithm is best-suited for I/O-bound processes? Support your answers here using specific simulation results.
2. For the SJF and SRT algorithms, what value of α produced the "best" results? Support your answer using specific simulation results.
3. For the SJF and SRT algorithms, how does changing from a non-preemptive algorithm to a preemptive algorithm impact your results? Are there specific examples of processes that have improved turnaround or wait times? Support your answers here using specific simulation results.
4. Identify at least three limitations of your simulation, in particular how the project specifications could be expanded to better model a real-world operating system. As an example, what other command-line arguments could you include (e.g., amount of time required to recalculate τ)?
5. Identify other "interesting" events and measurements that you could incorporate into your simulation.
6. Describe a priority scheduling algorithm of your own design, i.e., how could you calculate priority? What are advantages and disadvantages of your algorithm?

Again, the above questions are included here for you to think about and discuss with your teammates and/or others in the course. Feel free to discuss these on our Discussion Forum and/or continue this project beyond the end of this semester.