What is a scheduler and why is it important?

- We know context switch means removing a running process from a CPU and putting a new process in its place.
- But the question is, how do we know which process to select for running next? This is done by "scheduler".
- Scheduler brings order into the system and allow to optimize the system for a particular requirement (e.g. When Audio/Video processes are created, they get to run first)

How can we know if a scheduler (scheduling policy) is performing well?

- Different systems optimize for different metrics and so, systems tend to use different scheduling policies. e.g. Desktop is optimized for interactive applications but a real time system is different.
- A metric is decided based on the requirement. For a desktop system, response time could be the metric.
- Commonly used metrics:
  - Response time = First scheduled time - Arrival time.
  - Turnaround time = Completion time - Arrival time.
  - Fairness = If all active process receive equal amount of CPU time.

Classification of workloads

- Interactive (or short jobs): These are jobs that incur frequent I/Os. e.g. User typing in notepad. Delay between keys typed could be in the order of several milliseconds which is really high in the context of OS. These type of workloads prefer to optimize on response time.
- CPU - Intensive: Usually referred as batch jobs. Long running process without much I/O. These type of workloads prefer turaround time.

Scheduling policies

Assumptions

1. Run for same amount of time
2. Arrive at the same time
3. Run to completion
4. no IO
5. runtime is known

- FIFO
  - Simple
  - If assumption #1 is removed, affected by the problem of convoy-effect (or head-of-line blocking)

- SJF (Shortest Job First)
  - Handles convoy effect problem and better than FIFO in terms of turnaround time
If assumption #2 is removed, affected when short jobs come during runtime of a long job. Short jobs get stalled.

- **STCF (Shortest Time to Complete First)**
  - Check for shortest job on each process arrival (Assumption #3 relaxed - since we don't allow a process to run till completion) and perform the shortest job check
  - Running shortest jobs first leads to better turnaround time
  - Good for turnaround metric but poor for response time

- **RR (Round Robin)**
  - Run every active process for fixed period of time (called *schedule quantum*) in a cyclic order
  - Relies on timer interrupt to move onto next active process
  - Good for response time but poor for turnaround time

How to consider I/O activity while scheduling?

- If a process running on the CPU issues I/O, better utilization could be achieved if we schedule a different process
- Break process into subprocesses such that every CPU activity represents a process (since performing I/O does not need CPU execution we ignore that part of the process)
- Now perform STCF over the subprocesses - better utilization

Types of scheduler

- At a very high level, schedulers can be classified as preemptive and non-preemptive
  - Non-preemptive: A process runs until it yields the CPU by itself (performing I/O would result in yielding CPU)
  - Preemptive: A running process could be interrupted and switched with other process

Questions

1. What would be the metric for context switches?
2. Context switch : scheduling policy is an example of mechanism : policy. Think of other examples that form such a mechanism : policy pair in OS.
3. FIFO, SJF, STCF, RR - Identify if they are non-preemptive or preemptive.
4. Which policy (from the 4 that is discussed above) would provide better fairness?