Project Presentation
CS744 Autumn 2024

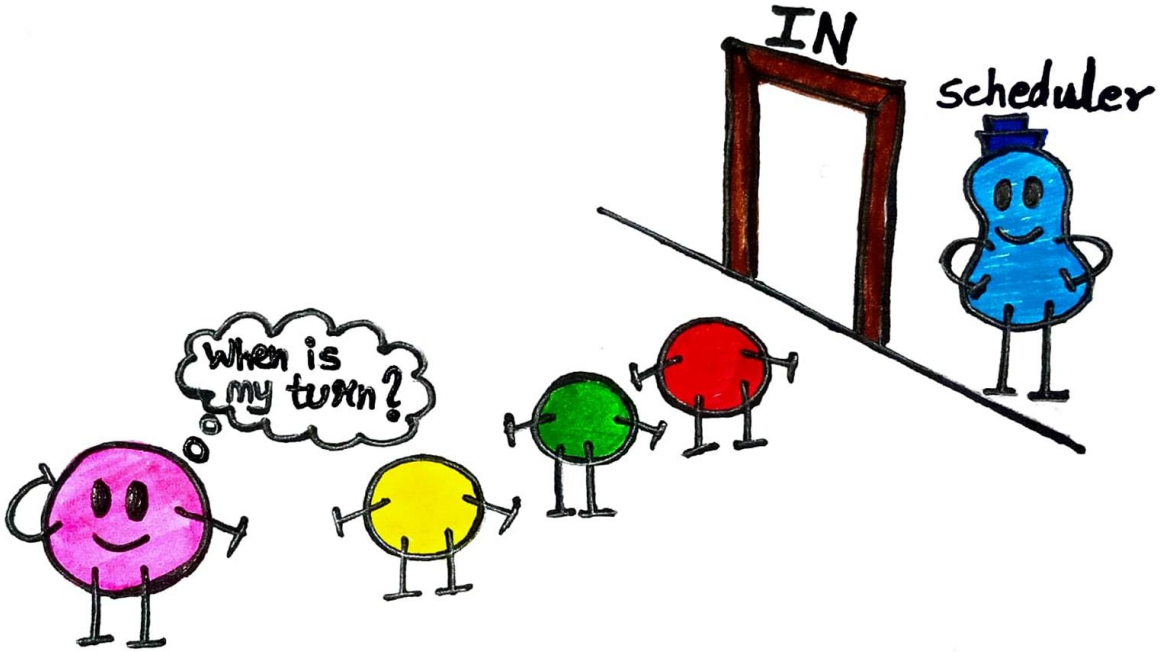
Scheduling Algorithms in xv6

Team Name: SnoopProc

Shalaka Thorat (24M0848)

Github Link:

25th November 2024


IN
scheduler
When is my turn?

Context

- Scheduling algorithms are essential in Operating Systems as they determine how CPU allocation to the processes, and in turn influence CPU utilization.
- This project provides a practical aspect of exploring scheduling algorithms by implementing them in xv6.
- This project aims to enhance the CPU scheduling mechanism in the xv6.

Problem Description

To implement a set of new scheduling algorithms and demonstrate their correctness and performance.

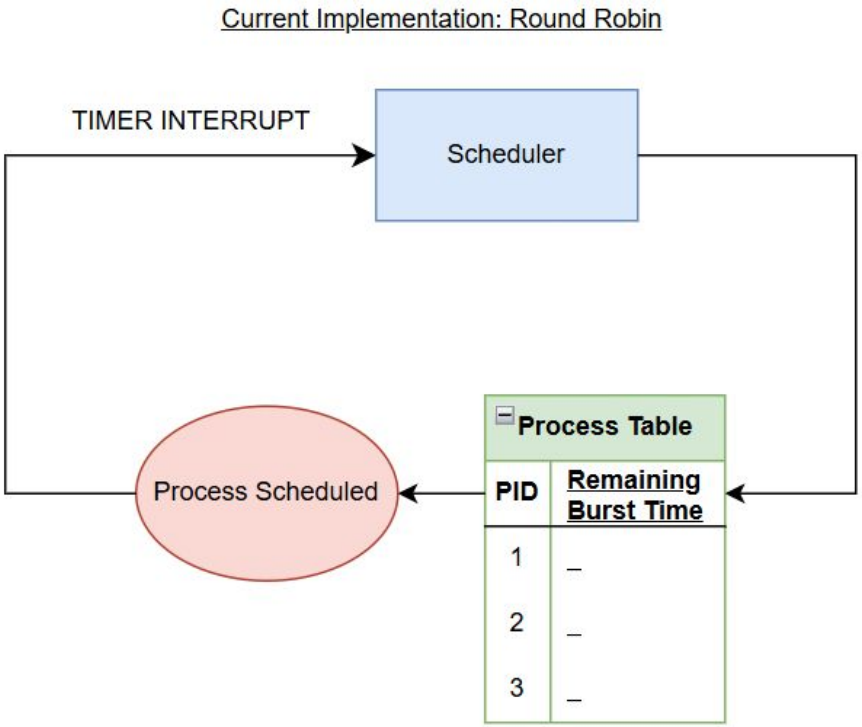
Pick different scenarios of processes so that you can explain which is the best scheduling policy for that situation.

Provide a system call interface to set scheduling hints per process.

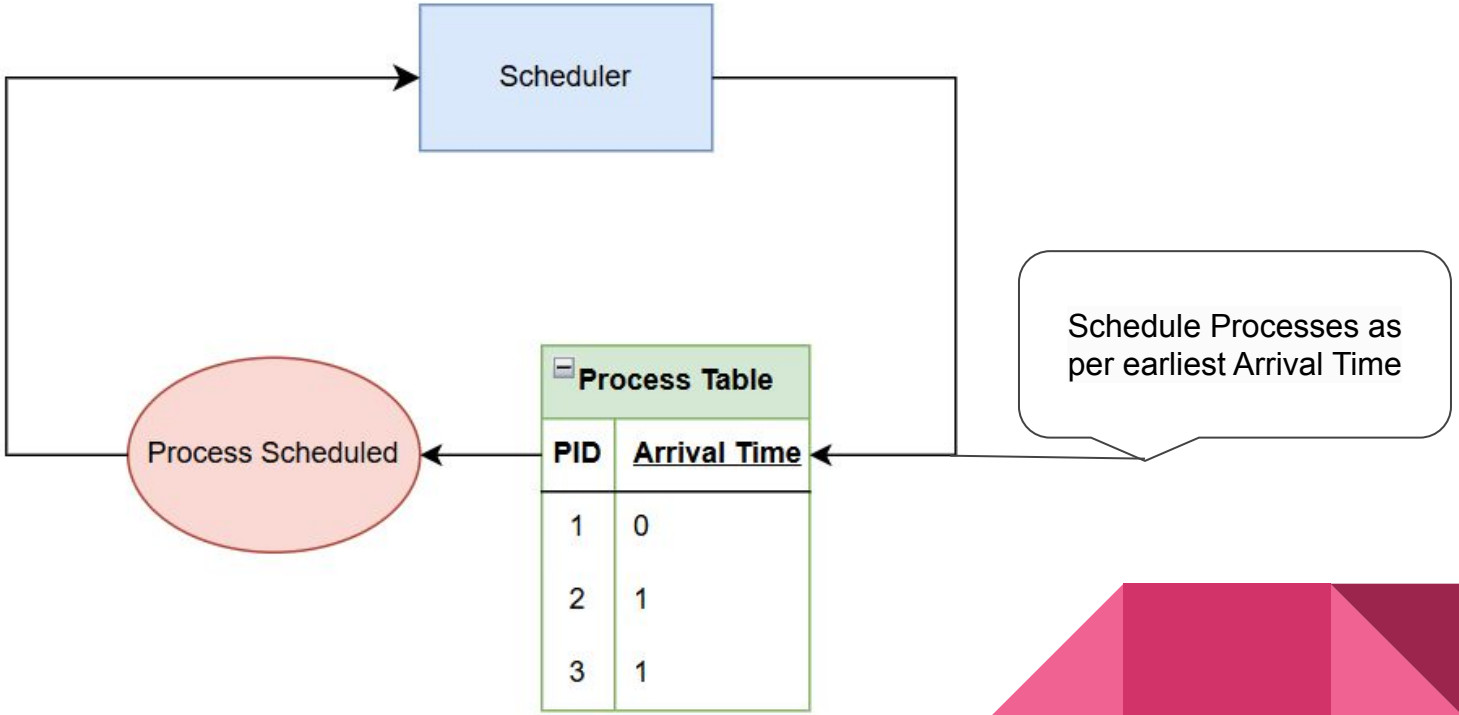
Components of the Project

1) Base OS xv6
    a) proc.c (Contains default scheduling algorithm logic)
    b) proc.h (Contains the metadata for a process)
2) Process Table and Process States
3) Scheduling Algorithms
    a) FCFS
    b) Priority Scheduling
    c) Priority Scheduling with Aging

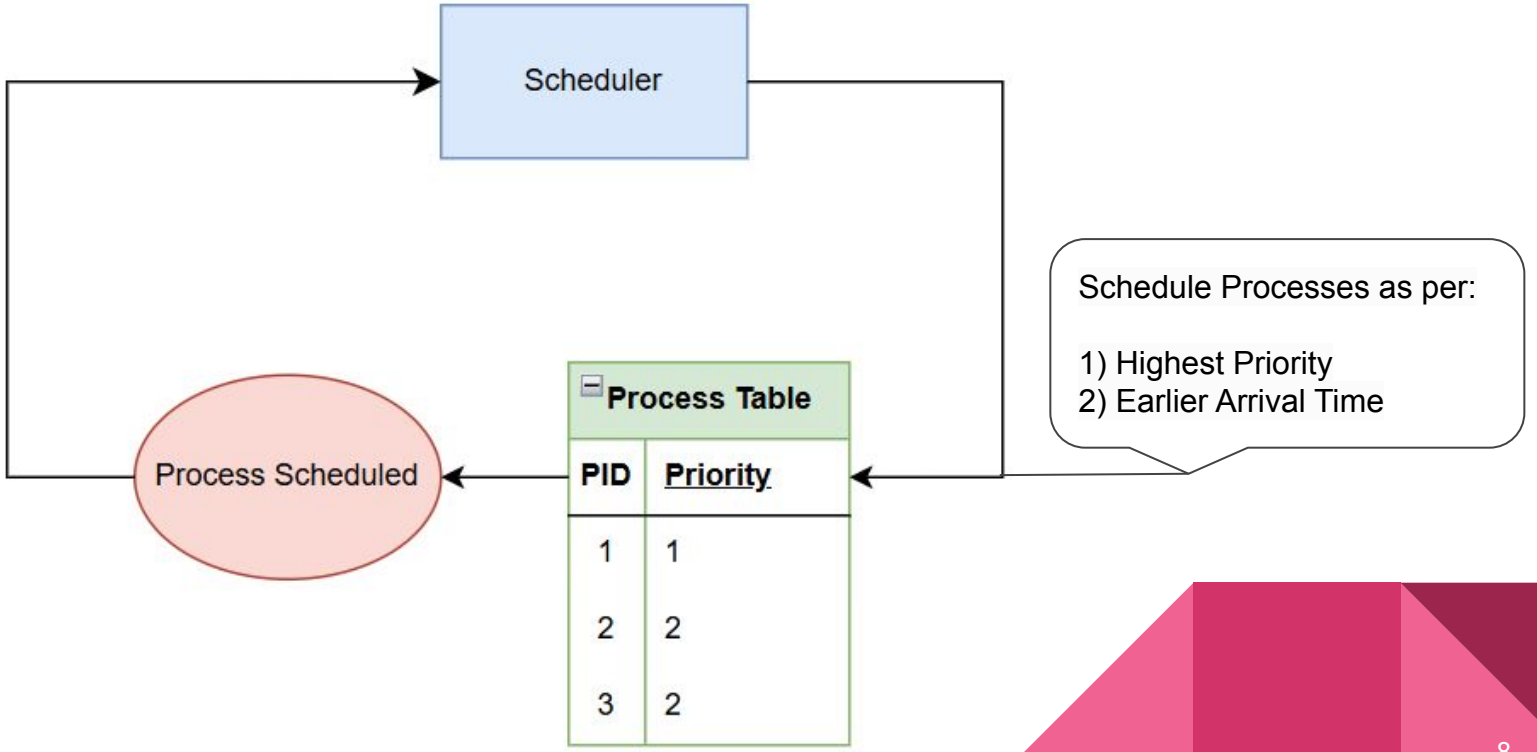
Current Design:



Design For FCFS:



Design for Priority Scheduling:



Implementation Details: FCFS

```
// Iterate over ptable and find the earliest process according to arrival_time
for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
{
    if(p->state != RUNNABLE)
            continue;

    if(earliest == 0 || p->arrival_time < earliest->arrival_time) // Checking if there is any earlier process
            earliest = p;
}
```

Implementation Details: Priority Scheduling

```
for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
{
  if(p->state != RUNNABLE)
    continue;

  if(high_priority == 0 || p->priority < high_priority->priority) // Checking if there is any higher priority process
            high_priority = p;
  else if(p->priority==high_priority->priority && p->arrival_time < high_priority->arrival_time)
            high_priority = p;
}
```

Implementation Details: Aging

```
  if(p->aging_time > AGING_LIMIT) // Adjusting Priority as per AGING_LIMIT
  {
     old_priority = p->priority;
     p->priority -= 1;
     if(p->priority < 0)
         p->priority = 0;
     p->aging_time = 0; // Resetting as process is now of higher priority -> more chances of getting selected

     cprintf("Priority Changed, PID: %d (%d) -> (%d)\n", p->pid, old_priority, p->priority);
  }

  if(high_priority == 0 || p->priority < high_priority->priority) // Checking if there is any higher priority process
            high_priority = p;
  else if(p->priority==high_priority->priority && p->arrival_time < high_priority->arrival_time)
            high_priority = p;
}
```

Evaluation

- Analyzing Scheduling Algorithms for a set of processes
- Varying Arrival Times of the Processes (Experiment 1)
- Varying Workload and Analyzing effect on the Scheduling Algorithms (Experiment 2)

Metrics Used: TAT

Analyzing Scheduling Algorithms for Processes

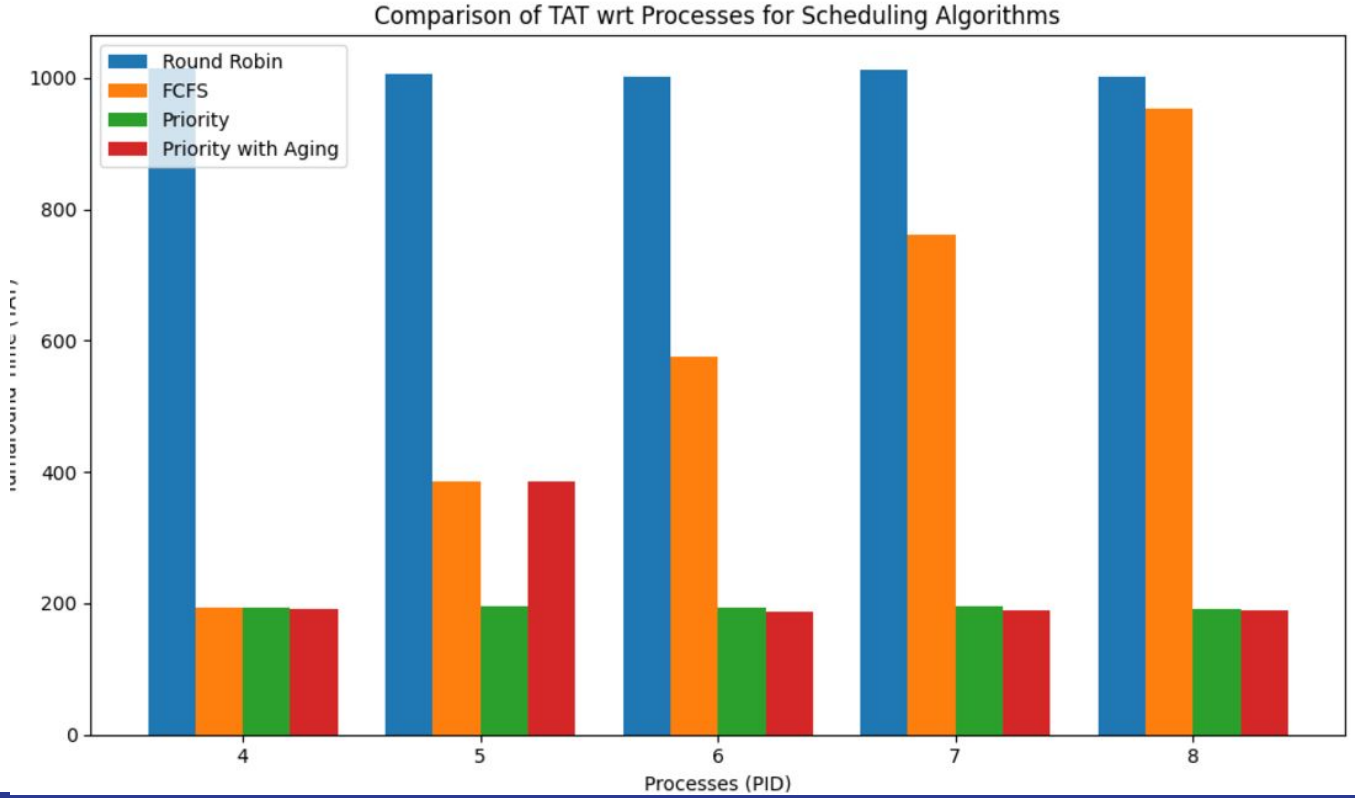
Observation:

Basic First hand Evaluation was to measure correctness of the Algorithms, which when tested and analyzed, were working as expected.

```
$ simulate
Process 0 (PID: 4) finished
PID: 4, AT: 401, CT: 619, TAT: 218
Process 1 (PID: 5) finished
PID: 5, AT: 401, CT: 851, TAT: 450
Process 2 (PID: 6) finished
PID: 6, AT: 402, CT: 1078, TAT: 676
Process 3 (PID: 7) finished
PID: 7, AT: 402, CT: 1309, TAT: 907
Process 4 (PID: 8) finished
PID: 8, AT: 402, CT: 1527, TAT: 1125
PID: 3, AT: 400, CT: 1527, TAT: 1127
$
```

```
$ simulate_priority
Process 0 (PID: 4) (Pri: 1) finished
PID: 4, AT: 679, CT: 870, TAT: 191
Process 2 (PID: 6) (Pri: 1) finished
PID: 6, AT: 679, CT: 1060, TAT: 381
Process 1 (PID: 5) (Pri: 2) finished
PID: 5, AT: 679, CT: 1253, TAT: 574
Process 4 (PID: 8) (Pri: 3) finished
PID: 8, AT: 679, CT: 1445, TAT: 766
Process 3 (PID: 7) (Pri: 4) finished
PID: 7, AT: 679, CT: 1638, TAT: 959
PID: 3, AT: 678, CT: 1638, TAT: 960
$
```

Varying Arrival Times of the Processes



Observations & Inference for Experiment 1:

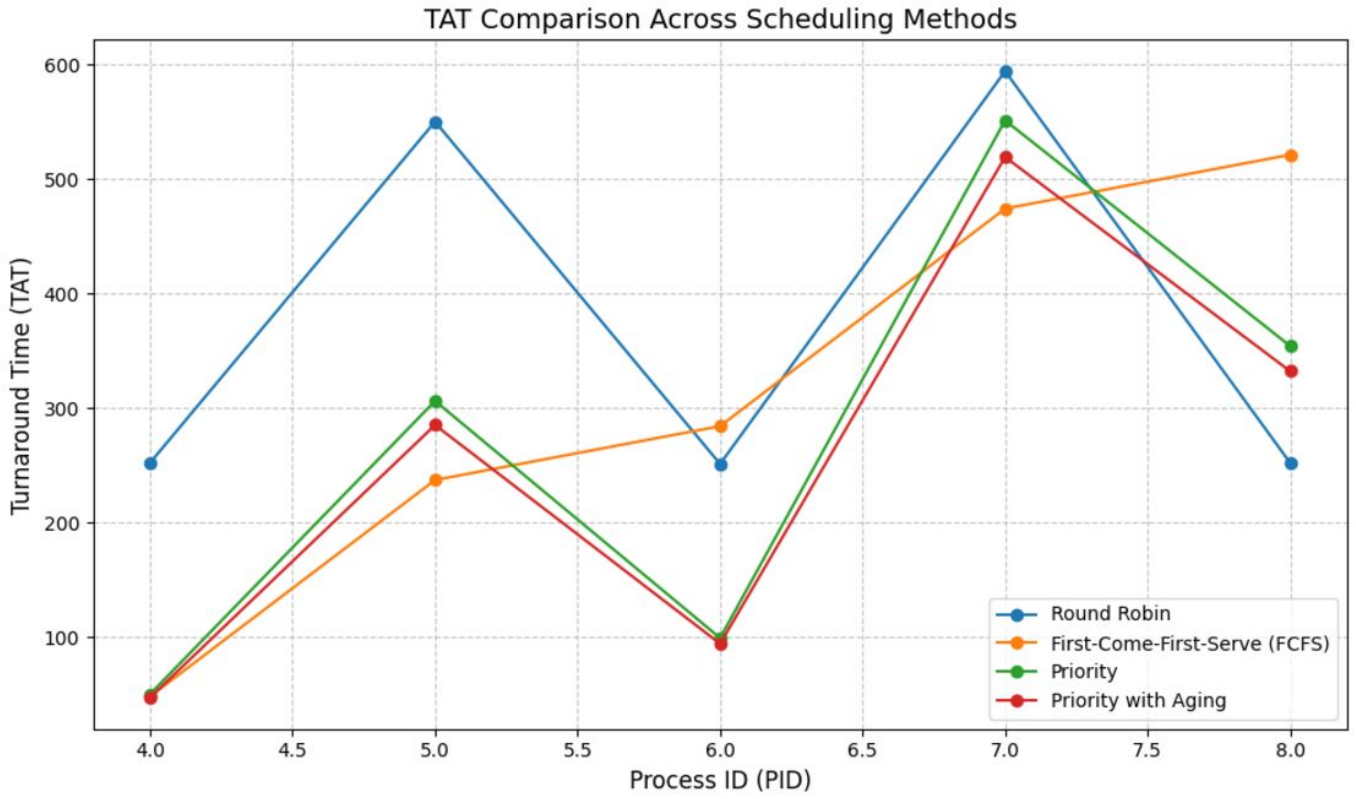
Observations:

1) Round Robin didn't have much effect for any parameter changes.
2) FCFS had cases of Starvation, while Priority Algorithm showed lesser Wait Time.

Inference:

If Arrival Times are aligned with their Priorities, along with Aging concept, processes would showcase lesser Wait Times.

Varying Workload for Processes



Observations & Inference for Experiment 2:

Observations:

1) Larger tasks required more TAT for Round Robin.
2) Lower Priority and Lower Workload processes suffered from Starvation.

Inference:

Assigning Priorities with respect to Workloads, will significantly result improvements in Wait Times.

Summary of Results

- For normal work cases: non-prioritized and similar workloads, Round Robin will outperform.

- For varied workloads, specific priority can be assigned with a view that higher workloads get higher priorities.
- Aging will backup the starved processes, resulting in lesser overall Wait Time.

Unfinished Scope

- Enabling Processes to be scheduled on different CPUs

- Dynamically calculate Burst-Time and try exploring more Algorithms

Challenges

- What is the current working of default scheduling in xv6
- What are the various functions being used, and their flow
- How to modify current algorithm, and practically implement other algorithms

Reflection

The chance to actually bring up theoretical concepts in practical manner was challenging and interesting.

This projects only acts as a base for exploring a handful of algorithms, but if provided a chance to redone, I would have tried to explore more on Burst Time calculation and implementing algorithms that wouldn’t be possible without burst time estimation.

Conclusion

This project explored various functions and flow of what actually happens, how processes are scheduled in xv6.

Also, it extends current implementation and tries to demonstrate on 2 different algorithms: FCFS, Priority Scheduling.

References

- https://www.geeksforgeeks.org/priority-cpu-scheduling-with-different-arrival-time-set-2/
- https://scholarworks.calstate.edu/downloads/hh63sx58j
- https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf