

Advanced Topics Final Presentation

By: Vishwesswaran Gopal

Introduction

- Fundamentals of Operating Systems
- **Course Content**
 - **Virtualization**
 - **Concurrency**
 - **Persistence**
 - **Security**
- **Course Goals: To understand how operating systems work.**
- **No** certification

Key Concepts

CPU Scheduling

- How can multiple processes (running programs) seemingly run on a single CPU, which can only perform one task at a time?
- **FIFO** (First In, First Out)
 - Strengths
 - Weaknesses

Virtualization of Memory

- How is physical memory split amongst many processes?
- *Base & Bounds*
 - Strengths
 - Weaknesses

Code for FIFO Scheduler

Implementation Code

```
6
7 class Node:
8     def __init__(self: Self, value: int, next: "Node" = None) -> None:
9         self.value = value
10        self._next = next
11
12        def __str__(self: Self) -> str:
13            return f"Node({self.value})"
14
15
16 class Queue:
17     def __init__(self: Self, node: "Node" = None) -> None:
18         self.first: "Node" = node
19         self.last: "Node" = node
20         self.add(node)
21
22     def add(self: Self, node: "Node") -> None:
23         if node is None:
24             return
25         if not isinstance(node, Node):
26             raise ValueError(
27                 "You passed in the wrong type for node, it should be an object of Node."
28             )
29         elif self.first is None:
30             self.first = node
31             self.last = node
32             return
33         prev: "Node" = self.last
34         self.last = node
35         prev._next = node
36
37     def pop(self: Self) -> "Node":
38         if self.first is None:
39             return
40         first: "Node" = self.first
41         self.first = first._next
42         return first
43
44     def __str__(self: Self) -> str:
45         current = self.first
46         nodes = []
47         while current:
48             nodes.append(str(current))
49             current = current._next
50         return f"Queue([{' -> '}.join(nodes)])"
```

Testing Code

```
53     q = Queue()  
54     q.add(Node(1))  
55     q.add(Node(2))  
56     print(q)  
57  
58     q.pop()  
59     print(q)  
60  
61     q.add(Node(3))  
62     print(q)  
63  
64     q.pop()  
65     q.pop()  
66     q.add(Node(4))  
67     print(q)
```



```
0ms: Queue([])
20ms: Queue([Process(1) -> Process(2)])
40ms: Queue([Process(2)])
60ms: Queue([Process(2) -> Process(3)])
80ms: Queue([Process(4)])
```

Terminal Output

The End!