

Write a program to implement Shortest Job First (SJF) Preemptive Scheduling for three processes and calculate the total context switches and average waiting time. The processes have burst times 10ns, 20ns, and 30ns, arriving at 0ns, 2ns, and 6ns, respectively.

```
kashish@LAPTOP-K3596DKL:~$ nano sjf_scheduling.py
kashish@LAPTOP-K3596DKL:~$ python3 sjf_scheduling.py
Process   Arrival   Burst    Waiting Time
P1         0         10       0ns
P2         2         20       8ns
P3         6         30       24ns

Total Context Switches: 2
Average Waiting Time: 10.67ns
kashish@LAPTOP-K3596DKL:~$ nano sjf_scheduling.py
kashish@LAPTOP-K3596DKL:~$ |
```

```
GNU nano 7.2 sjf_scheduling.py
class Process:
    def __init__(self, name, arrival_time, burst_time):
        self.name = name
        self.at = arrival_time
        self.bt = burst_time
        self.remaining_bt = burst_time
        self.completion_time = 0
        self.waiting_time = 0

def calculate_sjf_preemptive():
    processes = [Process("P1", 0, 10), Process("P2", 2, 20), Process("P3", 6, 30)]
    n = len(processes)
    current_time = 0
    completed = 0
    last_process = None
    context_switches = 0

    while completed < n:
        # Get processes that have arrived and not finished
        available = [p for p in processes if p.at <= current_time and p.remaining_bt > 0]

        if not available:
            current_time += 1
            continue

        # Pick the one with shortest remaining time
        current_p = min(available, key=lambda x: x.remaining_bt)
        # Track context switch
        if last_process is not None and last_process != current_p:
            context_switches += 1
        last_process = current_p
        current_p.remaining_bt -= 1
        current_time += 1
        if current_p.remaining_bt == 0:
            completed += 1
            current_p.completion_time = current_time
            # Waiting Time = Total Time - Arrival - Burst
            current_p.waiting_time = current_p.completion_time - current_p.at - current_p.bt

    avg_wt = sum(p.waiting_time for p in processes) / n
    return processes, context_switches, avg_wt

# Execution
results, switches, avg_wt = calculate_sjf_preemptive()
print(f"Processes: <10>{'Arrival':<10>{'Burst':<10>{'Waiting Time':<10>}}")
for p in results:
    print(f"{p.name:<10}{p.at:<10}{p.bt:<10}{p.waiting_time}ns")
print(f"\nTotal Context Switches: {switches}")
print(f"Average Waiting Time: {avg_wt:.2f}ns")
```

