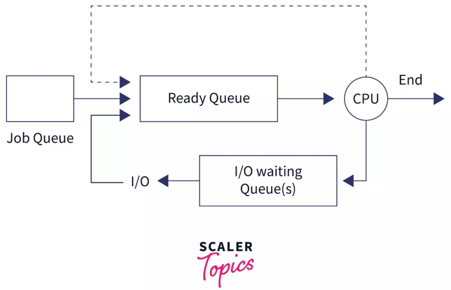
CPU SCHEDULING WINDOWS VS LINUX

Krish Sawant, Zohaib Quraishi, Lyles Williams

What is CPU Scheduling?



- Process used in OS to decide which process gets to use CPU next
- Minimize response and waiting time of a process
- Preemptive Scheduling
    a. OS can interrupt current process to have CPU handle another task
- Non-Preemptive Scheduling
    a. Once process starts it has to finish before CPU handles another task
- Maximize CPU usage
- Different CPU Scheduling algorithms
    a. RR(Round Robin)
    b. FIFO(First come first serve)
    c. SJF(Shortest Job First), etc
- Different OS like Windows and Linux have different algorithms for CPU scheduling

Why does CPU scheduling matter

- Affects how fast programs run and respond
- Ensures you are using your CPU efficiently
- Multiple processes which run at the same time have a fair and timely execution
- Ensures your hardware is being used to its fullest potential
- Makes sure hardware resources are not being wasted and are always doing something
- Without CPU Scheduling we could see things like
    - **Unoptimized servers**
    - **Inefficient CPU utilization**
    - **Hanging**
    - **A slow/unresponsive OS or program**
    - **A very high power consumption**
        i. **Leads to high costs which could hurt a business**
- Windows and Linux each have their own strengths in their schedulers

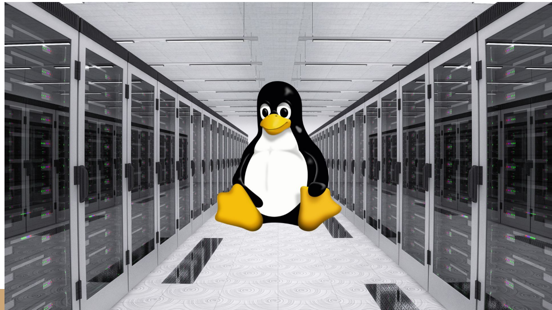
Real World Use Cases

Windows Strengths

- Ensures for a smooth gaming experience with minimal lag and/or frame drops.
- When watching Netflix, Hulu, etc we will see not much lag or a slow, unresponsive website UI because of other less important tasks
- When doing a lot of work where one of many tabs are needed, the user will not have to worry about the computer slowing down.
- Windows tries to focus on their user experience along with responsiveness of the system

Linux Strengths

- When programming, we want to make sure all processes have equal resources while still using CPU efficiently.
- When we are running virtual machines we make sure each containers uses same amount of resources
- For some systems where all processes should have equal resources we should use CFS
- Linux tries to focus more on things like programming, servers, etc.



Windows Scheduler

Priority-Based Round Robin

- Combines preemptive Round Robin scheduling with Priorities
- Use a Priority queue in order to track priorities and which process will be executed next
    a. **Priority of 1 means most important, the bigger the number, the less important**
- By combining preemptive Round Robin scheduling with Priorities we can achieve less average waiting time, average turnaround time and number of context switches
- Processes kept in increasing order or their remaining CPU burst time in the ready queue.
- New priorities are assigned according to the remaining CPU bursts of processes
- Process with shortest remaining CPU burst is assigned with highest priority

Round-Robin

i. We go through all processes giving each a time quantum and keep cycling

.

Time quantum is 5ms

Process Name	CPU burst time (ms)	Priority
A	22	4
B	18	2
C	9	1
D	10	3
E	4	5

Burst time and priority for different processes

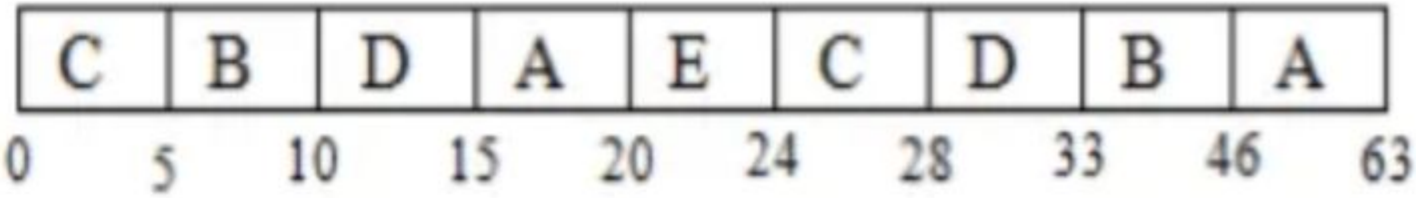
Table 2. Executed CPU burst for first round

S.No	Process	Executed Burst	Priority
1	C	5	1
2	B	5	2
3	D	5	3
4	A	5	4
5	E	4	5

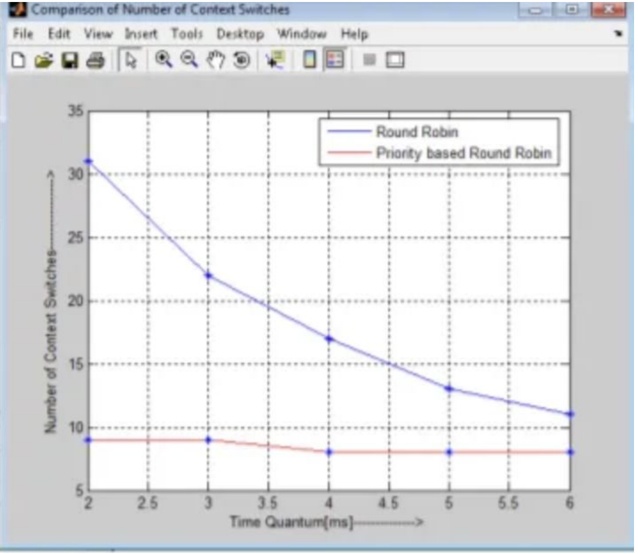
Priority associated with each process

Table 3. Remaining CPU burst for second round & new assigned priorities

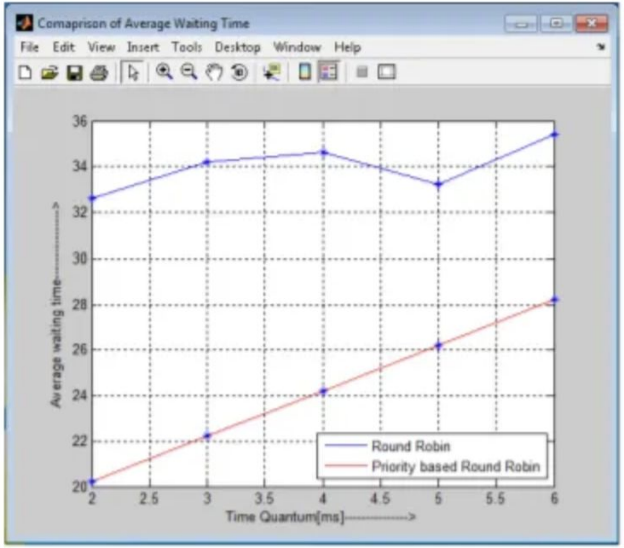
S.No	Process	Remaining Burst	Priority
1	C	4	1
2	D	5	2
3	B	13	3
4	A	17	4



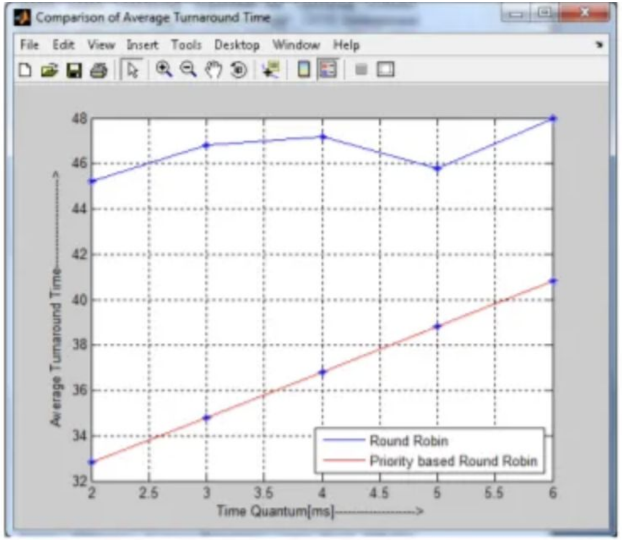
Gantt chart for proposed algorithm



Context Switches



Average Wait Time

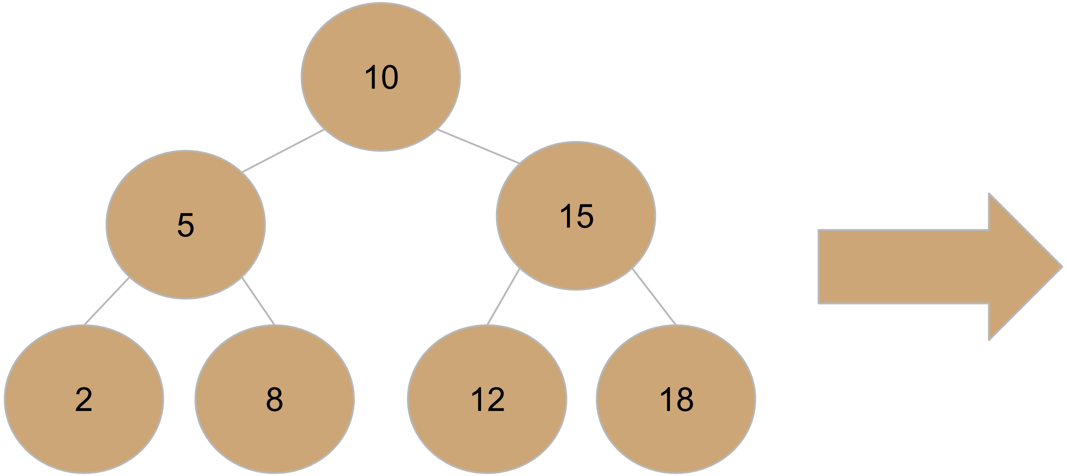


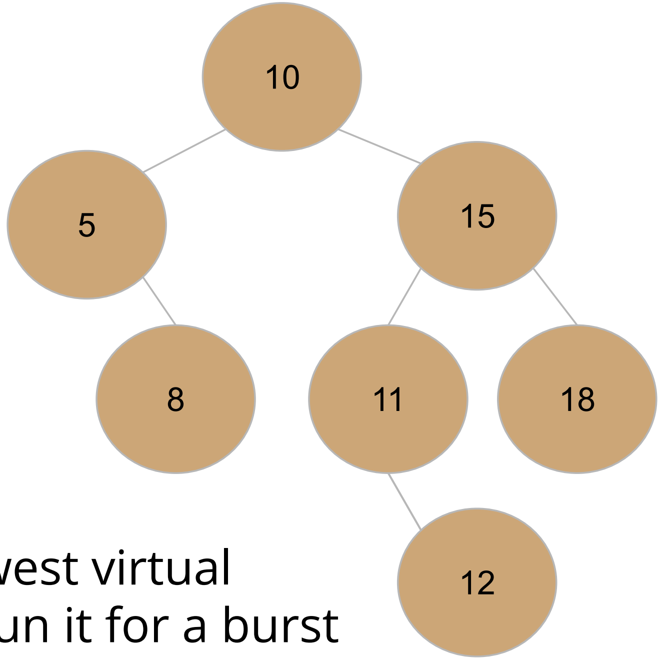
Average Turnaround Time

Linux Scheduler - CFS

- Default Linux Scheduler - Completely Fair Scheduler (CFS)
- Utilizes Red-black tree data structure
  - This keeps track of processes and CPU time
- Assigns CPU time to processes based on priority and amount of CPU time already used
- A general use scheduler
- Based on virtual runtime or ideal multitasking CPU
  - Virtual runtime is the amount of time a process has been running on the CPU
- Goal is to distribute all physical power and run each task at equal speeds in parallel
- Virtual runtime determines which process runs next.
  - Smallest virtual runtime runs next
  - Figured out through the red-black tree
- Nice values (from -20 to 19) determines which process gets priority to make things fair
  - -20 is the highest priority, 19 is the lowest
- Time Complexity: O(logn)

CFS Tree Example

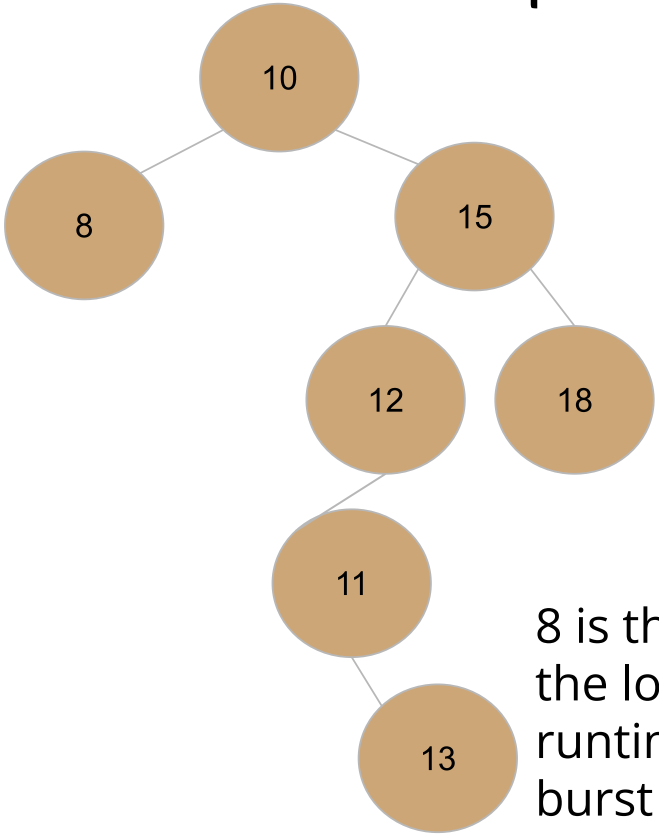




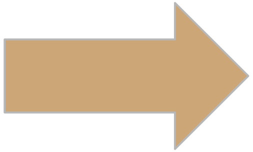
2 has the lowest virtual runtime so run it for a burst and remake the tree (bursts are 9 for this example)

5 has the lowest virtual runtime so run it for a burst and remake the tree (bursts are 9 for this example)

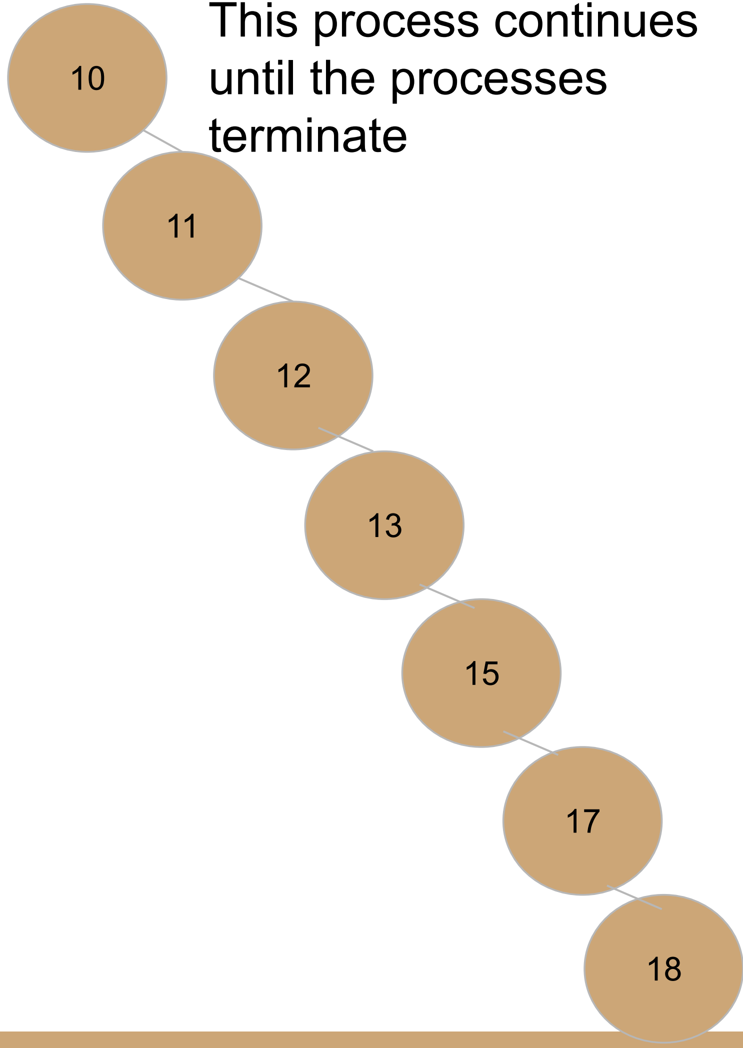
CFS Tree Example cont.



8 is the process with the lowest virtual runtime so run it for a burst (9)



This process continues until the processes terminate



Windows Scheduling vs Linux Scheduling Overview

	Windows	**Linux**
Scheduler Type	Priority Based Round Robin	Completely Fair Scheduler
Data Structure Used	Priority Queue	Red-black tree
Priority or Fairness	Priority Based	Fairness Based
Optimization	Optimized for responsiveness	General Use

Source Code Overview

- Simulated an environment where Priority Based Round Robin and Completely Fair Scheduler needed to run different processes of different priorities
- Kept track of variables like arrival, start, completion, waiting, response, average turn time, average wait time and average response time
- Made a Gantt chart

```
// ---------------------------
// Dummy workload
// ---------------------------
static Proc work[] = {
    // pid, arrival, burst, base_prio(0..15), nice(-20..19)
    {1,   0, 16, 10,   0}, // Medium priority → runs early but then yields
    {2,   2,  4,  8,  -5}, // Moderate priority → runs after higher ones
    {3,   4, 20,  6,   5}, // Lowest priority → runs last
    {4,   6,  3, 12, -10}, //High priority → short interactive job → runs early
    {5,  10, 12,  7,   0}, // Lower priority → runs later
    {6,  12,  8, 14,   2}, // Very high priority → runs early
};
```

Simulation Results - Windows

```
===== Windows-like (Priority RR) =====
pid  arrival burst  start  completion waiting  response    Priority
1    0       16     1      31         15       1           10
2    2       4      32     36         30       30          8
3    4       20     51     73         49       47          6
4    6       3      22     25         16       16          12
5    10      12     37     50         28       27          7
6    12      8      13     21         1        1           14
Makespan=73  CPU_util=0.863  AvgTurn=33.67  AvgWait=23.17  AvgResp=20.33
```

```
=== Gantt: Windows-like ===
 1 | ------------ P1 12
13 | --------- P6 21
22 | --- P4 25
26 | ----- P1 31
32 | ---- P2 36
37 | ---------- P5 46
47 | --- P5 50
51 | ---------- P3 60
61 | ---------- P3 70
71 | -- P3 73
```

- Low priority tasks wait longer
- Even if a high priority task arrives late it completes faster than low priority times
- Risk of starvation with the priority based round robin scheduling

Simulation Results - Linux

```
===== Linux CFS-like =====
pid  arrival burst  start  completion waiting  response          Priority
1    0       16     1      17         1        1                 10
2    2       4      18     22         1        16                8
3    4       20     34     73         36       30                6
4    6       3      38     41         21       32                12
5    10      12     28     61         32       18                7
6    12      8      23     53         28       11                14
Makespan=73  CPU_util=0.863  AvgTurn=38.83  AvgWait=19.83  AvgResp=18.00
```

```
=== Gantt: Linux CFS-like ===
  1 | ---------------- P1 17
 18 | ---- P2 22
 23 | ---- P6 27
 28 | ----- P5 33
 34 | --- P3 37
 38 | --- P4 41
 42 | ------ P3 48
 49 | ---- P6 53
 54 | -------- P5 61
 62 | ----------- P3 73
```

- CPU time distributed more evenly
- Waiting times are more fair
- High priority tasks are not prioritized

Simulation Results - Comparison & Conclusion

Comparison

- Linux CPU scheduling is more “fair”
- Windows works better if there are high priority tasks that need to be completed quickly

Conclusion

- Linux scheduling is more for general purpose while windows scheduling is more specialized

Work Cited

https://documentation.ubuntu.com/real-time/latest/explanation/schedulers/

https://www.geeksforgeeks.org/operating-systems/cpu-scheduling-in-operating-systems/

https://chiraggoyaliit.medium.com/priority-based-round-robin-cpu-scheduling-algorithm-f9aa8517a844