ELKOM (Jurnal Elektronika dan Komputer)
p-ISSN : 1907-0012 (print) e-ISSN : 2714-5417 (online)
Vol. 19, No. 1, Juli 2026
https://journal.stekom.ac.id/index.php/elkom

Analisis Komparatif Efisiensi Algoritma CPU Scheduling (FCFS, Round Robin, dan Priority) Berbasis Simulasi pada Lingkungan OS Simulator

Wildan Abu Bakar Sidiq[1*], Eksa Umar Gani[2], Djuniadi[3], Alfian Ardhiansyah[4]
[1, 2, 3,4]Department of Engineering, Universitas Negeri Semarang, Indonesia
Email author: wildanzstd@students.unnes.ac.id[1], eksaumar80@students.unnes.ac.id[2], djuniadi@mail.unnes.ac.id[3], alfianardhiansyah@mail.unnes.ac.id[4]

Article Info

Article history:

Received July9, 2026
Revised July11, 2026
Accepted July 13, 2026
Available July 14, 2026
Published July 30, 2026

Keywords:

Penjadwalan CPU
FCFS
Round Robin
Priority Scheduling
OS Simulator
Efisiensi Algoritma
Starvation

ABSTRAK

Penjadwalan CPU merupakan komponen krusial dalam sistem operasi modern untuk memastikan alokasi sumber daya komputasi yang efisien. Penelitian ini bertujuan untuk menganalisis dan membandingkan efisiensi kinerja tiga algoritma penjadwalan utama, yaitu First-Come First-Served (FCFS), Round Robin (RR) dengan time quantum 2 ms, dan Priority Scheduling (Non-Preemptive). Metode yang diterapkan adalah eksperimen berbasis simulasi menggunakan OS Simulator visual dengan dataset standar yang terdiri dari lima proses dengan variasi waktu kedatangan dan durasi eksekusi (burst time). Kinerja algoritma dievaluasi berdasarkan dua parameter utama: Average Waiting Time (AWT) dan Average Turnaround Time (ATAT). Hasil simulasi menunjukkan fenomena di mana FCFS dan Round Robin menghasilkan rata-rata waktu tunggu yang identik sebesar 7.8 ms, sedangkan Priority Scheduling mencatat kinerja terburuk dengan AWT 9.2 ms akibat terjadinya masalah starvation pada proses berprioritas rendah. Kesimpulannya, meskipun FCFS efisien untuk beban kerja sederhana, Round Robin terbukti lebih unggul dalam aspek keadilan distribusi waktu pemrosesan. Sebaliknya, penggunaan Priority Scheduling statis tidak disarankan tanpa penerapan mekanisme aging. Temuan ini memberikan validasi empiris bagi pengembang sistem dalam memilih strategi penjadwalan yang tepat sesuai karakteristik beban kerja.

Corresponding Author:

Wildan Abu Bakar Sidiq,
Universitas Negeri Semarang
Jl. Raya Banaran, Sekaran, Kec. Gn. Pati
Email: wildanzstd@students.unnes.ac.id

1. PENDAHULUAN

Unit Pemrosesan Pusat (*Central Processing Unit* atau *CPU*) merupakan komponen paling vital dalam arsitektur komputer yang berfungsi sebagai pusat eksekusi instruksi program. Dalam ekosistem sistem operasi modern yang mendukung multitasking, efisiensi penggunaan *CPU* sangat bergantung pada mekanisme penjadwalan *(scheduling)* yang diterapkan oleh sistem. Penjadwalan *CPU* memiliki tanggung jawab utama untuk memutuskan proses mana yang akan dialokasikan ke

pemroses ketika terdapat banyak tugas yang berada dalam antrean siap *(ready queue)*. Kegagalan dalam memilih algoritma penjadwalan yang tepat dapat berdampak fatal pada kinerja sistem secara keseluruhan, seperti meningkatnya waktu tunggu *(waiting time)*, rendahnya *throughput*, hingga terjadinya kemacetan pemrosesan [1]. Oleh karena itu, analisis efisiensi algoritma penjadwalan menjadi topik fundamental untuk memastikan optimalisasi sumber daya, terutama pada lingkungan komputasi modern yang memiliki karakteristik beban kerja sangat dinamis.

Sejumlah penelitian terdahulu telah banyak mengkaji kinerja berbagai algoritma penjadwalan *CPU* untuk berbagai skenario kasus. Penelitian terbaru [2] melakukan asesmen performa berbasis skenario dan menemukan bahwa *First-Come First-Served (FCFS)*, meskipun sederhana, sangat rentan terhadap fenomena convoy effect, di mana proses-proses kecil terhambat oleh proses besar yang memonopoli *CPU*. Di sisi lain, studi optimasi pada algoritma *Round Robin (RR)* [3] menunjukkan bahwa mekanisme pembagian waktu *(time quantum)* pada RR sangat efektif untuk keadilan, namun kinerjanya sangat sensitif; jika quantum tidak diatur dengan dinamis, *overhead* pada *context switching* akan meningkat drastis. Sementara itu, permasalahan pada *Priority Scheduling* dibahas dalam simulasi [4], yang menunjukkan risiko tinggi terjadinya *starvation* pada proses prioritas rendah jika tidak ditangani dengan baik. Analisis komparatif lainnya [5] yang diterbitkan tahun 2024 juga menegaskan bahwa tidak ada algoritma tunggal yang sempurna; pemilihan algoritma sangat bergantung pada apakah sistem mengutamakan keadilan waktu atau kecepatan penyelesaian tugas.

Berbagai penelitian terkini terus mengeksplorasi efektivitas algoritma penjadwalan pada lingkungan sistem operasi modern. Studi tinjauan sistematis oleh [6] serta analisis [7] pada lingkungan Linux menegaskan bahwa algoritma Round Robin memiliki keunggulan dalam keadilan pembagian waktu, namun kinerjanya sangat bergantung pada konfigurasi system. Hal ini didukung oleh [8] yang membandingkan Round Robin dengan Priority Preemptive, menemukan adanya trade-off signifikan antara waktu tunggu dan urgensi proses. Di sisi lain, implementasi algoritma Priority Scheduling semakin meluas ke berbagai domain, mulai dari penjadwalan tenaga medis rumah sakit [9] hingga optimasi basis data terdistribusi [10]. Kendati demikian, untuk mengatasi kelemahan statisnya, pendekatan prioritas dinamis (Dynamic Priority) mulai diadopsi untuk meningkatkan kualitas layanan pelanggan [11] dan keamanan lingkungan [12]. Bahkan, kombinasi hibrida antara Priority Scheduling dan Earliest Due Date (EDD) terbukti mampu meningkatkan akurasi jadwal produksi secara signifikan [13].

Tren penelitian terkini menunjukkan bahwa evaluasi algoritma penjadwalan tidak hanya relevan di level kernel sistem operasi, tetapi juga meluas ke manajemen antrean pada sistem IoT dan layanan publik. Analisis bibliometrik oleh [14] mengonfirmasi bahwa algoritma Round Robin masih menjadi fokus dominan riset karena karakteristik keadilannya. Hal ini dibuktikan dengan implementasi Round Robin pada perangkat IoT berdaya rendah untuk meminimalkan latensi komunikasi [15] serta efisiensi energi pada sistem pemantauan lingkungan [16]. Di sisi lain, prinsip antrean prioritas juga diadopsi secara luas untuk optimalisasi sistem informasi, seperti pada penjadwalan ujian akademik berbasis web [17] dan sistem administrasi kependudukan desa [18]. Bahkan, teori antrean dasar juga terbukti krusial dalam menganalisis efisiensi layanan kasir di sektor riil guna mengurangi waktu tunggu pelanggan [19].

Berdasarkan kesenjangan tersebut, penelitian ini bertujuan untuk melakukan analisis komparatif dan verifikasi empiris mengenai efisiensi algoritma *FCFS*, *Round Robin*, dan *Priority Scheduling* menggunakan simulator berbasis web *(OS Simulator)* yang lebih interaktif. Kebaruan *(novelty)* dari penelitian ini terletak pada penggunaan alat simulasi visual yang memungkinkan pemantauan *Gantt Chart* secara real-time untuk memvalidasi temuan-temuan teoritis di atas. Penelitian ini akan berfokus pada parameter *Average Waiting Time (AWT)* dan *Average Turnaround Time (ATAT)* untuk memberikan rekomendasi teknis bagi pengembang sistem dalam memilih strategi penjadwalan yang paling efisien.

2. METODE

Penelitian ini menerapkan metode eksperimental berbasis simulasi (*simulation-based experiment*) dengan pendekatan komparasi simultan. Metode ini dipilih untuk memastikan konsistensi lingkungan pengujian, di mana satu set data yang sama dieksekusi secara serentak oleh berbagai algoritma untuk meminimalkan bias variabel pengganggu.

2.1 Perangkat Simulasi

Alat utama yang digunakan adalah *OS Simulator* berbasis web yang dikembangkan secara open-source untuk memvisualisasikan penjadwalan CPU secara komprehensif [20]. Simulator ini memiliki karakteristik *multi-algorithm solver*, yang berarti sistem akan memproses input dataset menggunakan seluruh algoritma yang tersedia (termasuk *FCFS, SJF, Round Robin,* dan *Priority*) dalam satu kali eksekusi. Fitur ini memungkinkan perbandingan visual yang presisi melalui grafik batang *(bar chart)* gabungan yang dihasilkan secara otomatis.

Pendekatan simulasi interaktif berbasis web ini sejalan dengan metode yang dikembangkan oleh [21], di mana visualisasi algoritma preemptive dan non-preemptive terbukti efektif membantu pemahaman mendalam mengenai perilaku proses dan penggunaan sumber daya tanpa risiko merusak stabilitas sistem operasi utama.

2.2 Desain Skenario Uji (Dataset)

Agar perbandingan berjalan adil *(fair)*, penelitian ini menetapkan satu dataset standar yang dirancang untuk menciptakan kondisi persaingan sumber daya yang ketat. Dataset terdiri dari 5 proses dengan variasi *Burst Time* dan *Arrival Time* untuk menguji respons sistem terhadap antrean dinamis. Rincian dataset dapat dilihat pada Tabel 1.

Table 1. Dataset Skenario Pengujian Simulasi

Process ID	Arrival Time (AT)	Burst Time (BT)	Priority
P1	0	8	2
P2	1	4	1
P3	2	2	3
P4	3	1	4
P5	4	5	2

Keterangan: Pada kolom Priority, angka yang lebih kecil mengindikasikan prioritas yang lebih tinggi (1 = Tertinggi).

Parameter tambahan dikonfigurasi khusus untuk algoritma *Round Robin*, di mana nilai *Time Quantum (TQ)* ditetapkan sebesar 2 satuan waktu. Nilai ini dipilih untuk memaksa terjadinya *context switching* pada proses berdurasi panjang, sehingga dampak *overhead* algoritma dapat diobservasi sesuai prinsip kerja sistem time-sharing [22].

2.3 Prosedur Penelitian

Berbeda dengan pengujian manual yang dilakukan secara bertahap, penelitian ini menggunakan prosedur eksekusi simultan dengan langkah-langkah sebagai berikut:

a. Inisialisasi Data: Memasukkan parameter *Arrival Time*, *Burst Time*, dan *Priority* dari kelima proses ke dalam antarmuka simulator.
b. Konfigurasi Parameter: Menetapkan nilai *Time Quantum* = 2 ms pada pengaturan global simulator untuk kebutuhan perhitungan algoritma *Round Robin*.
c. Eksekusi Simultan: Menjalankan proses simulasi (mengklik tombol *Solve*). Simulator secara otomatis melakukan komputasi untuk seluruh jenis algoritma secara paralel.
d. Filtrasi dan Ekstraksi Data: Dari sekian banyak hasil algoritma yang ditampilkan (termasuk SJF dan Multilevel Queue), penelitian ini membatasi pengambilan data hanya pada tiga algoritma target:
    1) *First-Come First-Served* (FCFS).
    2) *Round Robin* (RR) dengan TQ=2.
    3) *Priority Scheduling* (Mode *Non-Preemptive*).
e. Validasi Visual: Mengambil tangkapan layar (*screenshot*) dari *Gantt Chart* dan tabel hasil spesifik untuk ketiga algoritma tersebut guna dianalisis pada bab pembahasan.

2.4 Metrik Evaluasi

Efisiensi algoritma diukur berdasarkan dua parameter utama yang dihitung menggunakan persamaan standar [1]:

a. Turnaround Time (TAT): Total waktu yang dihabiskan proses di dalam sistem (sejak datang hingga selesai), dihitung dengan Persamaan berikut:

$$TAT = Completion\ Time\ (CT) - Arrival\ Time\ (AT)$$

b. Waiting Time (WT): Total waktu proses menunggu di antrean siap (ready queue), dihitung dengan Persamaan berikut:

$$WT = Turnaround\ Time\ (TAT) - Burst\ Time\ (BT)$$

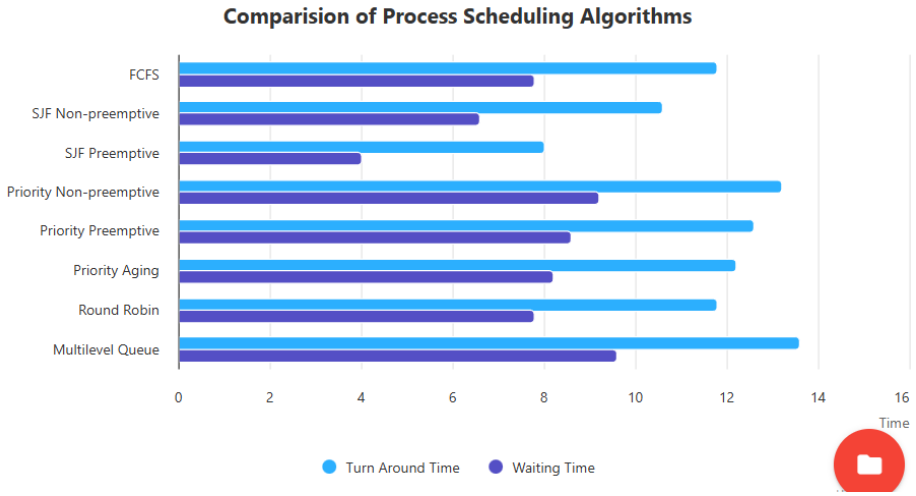
Nilai rata-rata atau *Average Waiting Time* (AWT) dari kelima proses akan menjadi indikator utama. Algoritma dengan AWT terendah akan dikategorikan sebagai algoritma paling efisien untuk dataset yang diujikan.

3. HASIL DAN PEMBAHASAN

Bagian ini memaparkan data kuantitatif yang diperoleh dari eksperimen simulasi serta memberikan analisis mendalam mengenai pola efisiensi yang ditemukan pada algoritma First-Come First-Served (FCFS), Round Robin (RR), dan Priority Scheduling.

3.1. Hasil Simulasi

Berdasarkan eksekusi simulasi menggunakan dataset yang dirancang pada Tabel 1, simulator menghasilkan visualisasi kinerja secara simultan untuk ketiga algoritma. Data visual perbandingan kinerja ditampilkan pada Grafik Batang (Bar Chart) di Gambar 1.



Gambar 1. Perbandingan Algoritma Proccess Scheduling

Dari grafik tersebut, terlihat pola visual di mana batang indikator untuk FCFS dan Round Robin memiliki ketinggian yang setara, sedangkan algoritma Priority Scheduling (Non-Preemptive) menunjukkan batang tertinggi, mengindikasikan kinerja waktu yang paling lambat dibandingkan dua algoritma lainnya.

Secara spesifik, rincian distribusi waktu eksekusi untuk algoritma Round Robin dengan parameter Time Quantum (TQ) sebesar 2 ms dapat dilihat pada Gambar 2.

Round Robin

Process ID	Burst time	Arrival time	Waiting Time	Turn Around Time	Completion time
1	8	0	12	20	20
2	4	1	4	8	9
3	2	2	7	9	11
4	1	3	8	9	12
5	5	4	8	13	17

Gambar 2. Rincian Waktu Eksekusi Proses pada Algoritma Round Robin

Untuk memberikan perbandingan yang presisi, data Average Waiting Time (AWT) dan Average Turnaround Time (ATAT) dari ketiga algoritma tersebut dirangkum dalam Tabel 2.

Table 2. Rekapitulasi Perbandingan Efisiensi Algoritma

Algoritma	Avg. Waiting Time (ms)	Avg. Turnaround Time (ms)
FCFS	7.8	11.8
Round Robin (TQ=2)	7.8	11.8
Prority (Non-Preemptive)	9.2	13.2

3.2. Pembahasan

Hasil eksperimen menunjukkan fenomena menarik di mana algoritma sederhana FCFS mampu menyamai kinerja rata-rata algoritma kompleks Round Robin, sementara Priority Scheduling mengalami degradasi kinerja signifikan. Berikut adalah analisis mendalam berdasarkan landasan teori.

a. Analisis Kinerja FCFS dan Efek Konvoi

Pada algoritma FCFS, proses P1 dengan burst time panjang (8 ms) datang paling awal pada t=0 dan langsung memonopoli CPU hingga selesai. Hal ini menyebabkan proses P2, P3, P4, dan P5 terpaksa mengantre di belakang P1. Kondisi ini mengonfirmasi teori mengenai Convoy Effect, sebuah situasi di mana proses-proses pendek terhambat oleh satu proses panjang di depan antrean [2]. Meskipun terjadi penumpukan, nilai rata-rata AWT FCFS terjaga di angka 7.8 ms. Hal ini disebabkan karena proses P3 (2 ms) dan P4 (1 ms) memiliki durasi eksekusi yang sangat singkat. Begitu P1 dan P2 selesai, P3 dan P4 dieksekusi dengan sangat cepat, sehingga menurunkan pembagi rata-rata total waktu tunggu.

b. Analisis Round Robin dan Aspek Keadilan (Fairness)

Temuan paling signifikan dalam simulasi ini adalah kesamaan nilai AWT antara Round Robin dan FCFS (7.8 ms). Secara teoritis, RR dirancang untuk meminimalkan waktu respons, namun pada kasus uji ini, nilai rata-ratanya tidak lebih baik dari FCFS. Mengacu pada penelitian [3], hal ini terjadi karena penggunaan Time Quantum statis (2 ms). Proses panjang P1 mengalami pemotongan (preemption) berulang kali, yang menambah overhead sistem. Studi terbaru oleh [23] menegaskan bahwa korelasi antara ukuran kuantum (quantum size) dan kinerja Round Robin sangat signifikan; nilai kuantum yang terlalu kecil, seperti pada skenario ini, akan meningkatkan frekuensi context switching secara drastis. Sebaliknya, kuantum yang optimal diperlukan untuk menyeimbangkan antara response time dan throughput. Meskipun rata-rata waktunya sama, Round Robin terbukti lebih unggul dalam aspek keadilan (fairness) sebagaimana dijelaskan oleh [5]. Pada FCFS, proses P4 harus menunggu 13 ms untuk berjalan. Sedangkan pada Round Robin (lihat Gambar 2), P4 hanya menunggu 8 ms karena mendapatkan jatah CPU lebih awal melalui mekanisme rotasi. Ini membuktikan RR lebih responsif untuk sistem interaktif.

c. Masalah Starvation pada Priority Scheduling

Algoritma Priority Scheduling mencatat kinerja terburuk dengan AWT 9.2 ms. Analisis jejak eksekusi menunjukkan bahwa hal ini disebabkan oleh mekanisme prioritas statis Non-Preemptive. Proses P1 dan P5 memiliki prioritas tinggi (2) dan durasi panjang, mendominasi penggunaan CPU. Permasalahan fatal terjadi pada proses P3 dan P4. Meskipun keduanya sangat singkat (hanya butuh 1-2 ms), mereka memiliki prioritas rendah (3 dan 4). Akibatnya, P3 dan P4 dipaksa "mengalah" dan menunggu seluruh proses berprioritas tinggi selesai. Fenomena ini memvalidasi temuan [4] mengenai risiko starvation (kelaparan sumber daya). Tanpa mekanisme aging, proses kecil berprioritas rendah akan mengalami penundaan eksekusi drastis, yang secara langsung melambungkan nilai rata-rata waktu tunggu sistem.

d. Implikasi Pemilihan Algoritma

Berdasarkan data komparatif di atas, dapat disimpulkan bahwa untuk beban kerja campuran statis di mana proses panjang datang di awal, FCFS masih menjadi pilihan efisien karena minim overhead. Namun, jika tujuan sistem adalah menjamin responsivitas bagi seluruh pengguna (agar tidak ada satu proses pun yang menunggu terlalu lama), Round Robin adalah pilihan mutlak. Sebaliknya, Priority Scheduling tanpa modifikasi dinamis terbukti tidak efisien untuk dataset ini karena menyebabkan inefisiensi pada proses-proses kecil.

Meskipun dalam eksperimen ini FCFS dan Round Robin menunjukkan rata-rata waktu tunggu yang setara, studi komparatif oleh [24] menyoroti bahwa Preemptive Shortest Job First (SJF) sering kali lebih unggul secara matematis dalam meminimalkan waiting time, meskipun memiliki kompleksitas implementasi yang lebih tinggi. Ke depan, arah optimasi penjadwalan diprediksi akan beralih ke pendekatan cerdas, seperti integrasi Machine Learning (misalnya Random Forest Regression) untuk memprediksi burst time proses secara akurat guna mengatasi kelemahan algoritma klasik, sebagaimana diusulkan oleh [25].

4. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan analisis komparatif dan hasil simulasi yang telah dilakukan menggunakan OS Simulator terhadap algoritma FCFS, Round Robin, dan Priority Scheduling, dapat ditarik beberapa kesimpulan sebagai berikut:

a. Ekuivalensi Waktu Tunggu: Pada dataset pengujian dengan variasi burst time campuran, algoritma FCFS dan Round Robin (TQ=2) menghasilkan rata-rata waktu tunggu (Average Waiting Time) yang identik, yaitu 7.8 ms. Hal ini membuktikan secara empiris bahwa algoritma kompleks seperti Round Robin tidak selalu menjamin penurunan waktu tunggu rata-rata

dibandingkan algoritma sederhana jika parameter Time Quantum tidak dioptimalkan secara dinamis terhadap beban kerja.

b. Keunggulan Keadilan (Fairness): Meskipun nilai rata-ratanya sama, Round Robin terbukti jauh lebih unggul dalam aspek keadilan distribusi sumber daya. Visualisasi Gantt Chart menunjukkan bahwa Round Robin mampu membagi jatah CPU secara merata, sehingga proses dengan burst time pendek tidak mengalami penundaan ekstrem seperti yang terjadi pada FCFS (convoy effect).
c. Kelemahan Prioritas Statis: Algoritma Priority Scheduling (Non-Preemptive) mencatat kinerja terburuk dengan rata-rata waktu tunggu 9.2 ms. Kinerja buruk ini disebabkan oleh terjadinya starvation relatif pada proses-proses berdurasi pendek yang kebetulan memiliki prioritas rendah, yang dipaksa menunggu selesainya eksekusi proses panjang berprioritas tinggi.

4.2 Saran

Untuk pengembangan penelitian selanjutnya dan implementasi teknis, penulis menyarankan hal-hal berikut:

a. Optimasi Time Quantum: Disarankan untuk meneliti penggunaan algoritma Round Robin dengan penentuan Time Quantum dinamis (misalnya menggunakan metode mean atau median dari burst time) untuk menyeimbangkan antara overhead dan responsivitas.
b. Mekanisme Aging: Pada implementasi Priority Scheduling, wajib diterapkan mekanisme aging (peningkatan prioritas secara bertahap seiring lamanya waktu tunggu) untuk mencegah terjadinya starvation.
c. Skala Pengujian: Penelitian selanjutnya dapat memperluas cakupan dengan menggunakan dataset acak (random generated) dalam jumlah masif (di atas 50 proses) untuk menguji stabilitas algoritma pada kondisi beban puncak (peak load).

REFERENSI

[1] H. Omar, K. Jihad, and S. F. Hussein, "Comparative analysis of the essential CPU scheduling algorithms," *Bull. Electr. Eng. Informatics*, vol. 10, pp. 2742–2750, Oct. 2021.

[2] O. Hajjar, E. Mekhallalati, N. Annwty, F. Alghayadh, I. Keshta, and M. Algabri, "Performance Assessment of CPU Scheduling Algorithms: A Scenario-Based Approach with FCFS, RR, and SJF," *J. Comput. Sci.*, vol. 20, no. 9, pp. 972–985, 2024, doi: 10.3844/jcssp.2024.972.985.

[3] D. Biswas, M. Samsuddoha, M. R. Al Asif, and M. M. Ahmed, "Optimized Round Robin Scheduling Algorithm Using Dynamic Time Quantum Approach in Cloud Computing Environment," *Int. J. Intell. Syst. Appl.*, vol. 15, no. 1, pp. 22–34, 2023, doi: 10.5815/ijisa.2023.01.03.

[4] T. D. Putra and R. Purnomo, "Simulation of Priority Round Robin Scheduling Algorithm," *Sink. J. dan Penelit. Tek. Inform.*, vol. 6, no. 4 SE-, pp. 2170–2181, Oct. 2022, doi: 10.33395/sinkron.v7i4.11665.

[5] W. Pan, "Comparison and Analysis of Scheduling Algorithms: Exploring Performance, Time, Fairness, and Applicable Scenarios," *Highlights Sci. Eng. Technol.*, vol. 120 SE-, pp. 1–7, doi: 10.54097/4asdtt07.

[6] I. Fadhilah and H. Siregar, "Systematic Literature Review : Perbandingan Algoritma Round Robin dan Shortest Job First dalam Penjadwalan CPU," *BIOS J. Teknol. Inf. dan Rekayasa Komput.*, vol. 6, no. 2, pp. 74–83, 2025, [Online]. Available: https://bios.sinergis.org/bios/article/view/168

[7] A. I. Azzam and H. Siregar, *Analisis Perbandingan Algoritma Penjadwalan CPU pada Sistem Operasi Linux*, vol. 9, no. 3 SE-. 2025, pp. 818–825. doi: 10.33395/remik.v9i3.14924.

[8] W. Widiarto, R. A. Chaerunnisa, R. A. Tsaqif, and S. S. Nadia, "Analisis Perbandingan Penjadwalan Proses Menggunakan Algoritma Round Robin dan Priority Preemptive," *Progresif J. Ilm. Komputer; Vol 20, No 2 Agustus 2024*, Aug. 2024, [Online]. Available: https://ojs.stmik-banjarbaru.ac.id/index.php/progresif/article/view/2169

[9] Tati Harihayati Mardzuki, R. Lubis, and M. Abdulloh, "Design of Nursing Staff Scheduling System for Hospitals Using Priority Scheduling Algorithm," *Komputa J. Ilm. Komput. dan Inform.*, vol. 14, no. 1 SE-Articles, pp. 108–119, May 2025, doi: 10.34010/komputa.v14i1.15948.

[10] M. Mutasar and S. Rahmah, "OPTIMASI BASIS DATA TERDISTRIBUSI DENGAN ALGORITMA PRIORITY SCHEDULING," Aug. 2024.

[11] M Fahri Aditya Nasution, Suendri, and A. Muliani Harahap, "Customer Service Information System Using Dynamic Priority Scheduling Algorithm At PT Sumatra Sistem Integrasi," *J. Inf. Syst. Technol. Res.*, vol. 2,

no. 1 SE-Articles, pp. 25–37, Jan. 2023, doi: 10.55537/jistr.v2i1.324.
[12] F. Masyfa, D. Kartikasari, and Tibyani, "Penjadwalan dan Pelaporan Menggunakan Dynamic Priority Scheduling dan Geolocation untuk Keamanan Lingkungan Scheduling and Reporting using Dynamic Priority Scheduling and Geolocation for Environmental Security," *Februari*, vol. 22, no. 1, pp. 195–206, 2023.
[13] M. I. Afrianto, F. Fauziah, and Y. F. Wijaya, "Kombinasi Algoritma Priority Scheduling dan Earliest Due Date untuk Sistem Penjadwalan Slitting Produk Berbasis Web," *Teknokom*, vol. 7, no. 1, pp. 180–186, 2024, doi: 10.31943/teknokom.v7i1.176.
[14] D. N. Al Husaeni and J. Kusnendar, "Analisis Trend Penelitian Penggunaan Algoritma Penjadwalan serta Faktor yang Mempengaruhinya: Analisis Bibliometrik R dan Pemetaan VOSviewer," *J. Nas. Teknol. dan Sist. Inf.*, vol. 11, no. 1, pp. 57–66, 2025, doi: 10.25077/teknosi.v11i01.2025.57-66.
[15] B. Septian, A. Adrianda, M. F. Ridho, and M. Misbahuddin, "An IoT-Enabled Low Latency Automatic Identification System Using Round-Robin Scheduling Algorithm," *J. Artif. Intell. Softw. Eng.*, vol. 5, no. 2, pp. 403–409, 2025, doi: 10.30811/jaise.v5i2.6512.
[16] A. E. Burhandenny, S. Pranoto, and D. Suprihanto, "Optimal Scheduling of IoT Devices Using Round Robin Algorithm in Environmental Monitoring Systems: A Simulation Approach," *Ajie*, vol. 9, no. May, pp. 84–95, 2025, doi: 10.20885/ajie.vol9.iss2.art2.
[17] M. I. Yanwari, A. S. Prabuwono, T. R. Yudantoro, N. B. Aji, Wiktasari, and S. Handoko, "Priority Scheduling Implementation for Exam Schedule," *Indones. J. Inf. Syst.*, vol. 5, no. 2, pp. 80–89, 2023, doi: 10.24002/ijis.v5i2.6871.
[18] A. A. Rohmah and D. Gunawan, "Implementasi Algoritma Priority Scheduling Sistem Informasi Pelayanan Administrasi Kependudukan Desa," *J. Inform. J. Pengemb. IT*, vol. 8, no. 3, pp. 181–187, 2023, doi: 10.30591/jpit.v8i3.4891.
[19] T. N. Naranata and S. Sunarso, "Analisis Sistem Antrean Dalam Optimalisasi Pelayanan Kasir Hotway's Chicken Di Solo," *EKOMA J. Ekon. Manajemen, Akunt.*, vol. 5, no. 1, pp. 492–511, 2025, [Online]. Available: https://ulilalbabinstitute.id/index.php/EKOMA/article/view/11741
[20] S. Mishal, "OS Simulator: A Web-Based Operating System Scheduling Visualizer," 2024, *https://github.com*. [Online]. Available: https://github.com/mishal23/os-simulator
[21] G. A. Rumahorbo, Zulfahmi Indra, Alfarizi Wijaya, Melika Debiyana Putri, and C. S. Buulolo, "Analisis Perbandingan Algoritma Penjadwalan Prioritas Preemptive dan Non-Preemptive Menggunakan Aplikasi Web Interaktif," *Tek. J. Ilmu Tek. dan Inform.*, vol. 5, no. 2, pp. 150–158, 2025, doi: 10.51903/teknik.v5i2.994.
[22] G. G. Abraham Silberschatz, Peter B. Galvin, *Operating System Concepts*, 10th ed. Wiley, 2021.
[23] F. Anjany, S. Fikroh Al Kaamil, and M. Ainul Yaqin, "Hubungan Antara Ukuran Kuantum (Quantum Size) dan Kinerja Algoritma Round-Robin (Studi Kasus Penjadwalan CPU)," *J. Pustaka Data (Pusat Akses Kaji. Database, Anal. Teknol. dan Arsit. Komputer)*, vol. 5, no. 2 SE-Artikel, pp. 283–294, Dec. 2025, doi: 10.55382/jurnalpustakadata.v5i2.1395.
[24] R. Purnomo and T. D. Putra, "Comparative Study: Preemptive Shortest Job First and Round Robin Algorithms," *Sink. J. dan Penelit. Tek. Inform.*, vol. 8, no. 2 SE-, pp. 756–763, Mar. 2024, doi: 10.33395/sinkron.v8i2.12525.
[25] Aditya Putra Ramdani, Achmad Solichan, Basirudin Ansor, Muhammad Zainudin Al Amin, Nova Christina Sari, and Kilala Mahadewi, "Optimizing Shortest Job First (SJF) Scheduling through Random Forest Regression for Accurate Job Execution Time Prediction," *Int. J. Inf. Eng. Sci.*, vol. 1, no. 3, pp. 39–49, 2024, doi: 10.62951/ijies.v1i3.138.