LINIER
LITERATUR INFORMATIKA & KOMPUTER
e-ISSN: 3063-2218
DOI : 10.33096/linier.v3i2.3643
Literatur Informatika & Komputer
Vol 3, No 2 (2026), pp 260-270

Analisis Perbandingan Algoritma First Come First Served dan Round Robin Pada Proses Penjadwalan

Aditya Ramadan[a], Siska Anraeni[b], Sugiarti[c]
Universitas Muslim Indonesia, Makassar, Indonesia
[a]13020190053@umi.ac.id; [b]siska.anraeni@umi.ac.id; [c]sugiarti.sugiarti@umi.ac.id

Received: 12-08-2025 | Revised: 23-05-2026 | Accepted: 26-06-2026 | Published: 29-06-2026

Abstrak

Masalah yang sering muncul di proses penjadwalan dalam sistem operasi sering menghadapi berbagai masalah yang dapat mempengaruhi kinerja dan efisiensi. Tujuan dari penelitian ini adalah untuk menganalisis dan membandingkan kinerja dua algoritma penjadwalan, yaitu FCFS dan RR, guna menentukan algoritma yang lebih optimal dalam mengelola proses multitasking. Metode yang digunakan meliputi studi literatur, perancangan simulasi menggunakan 10 proses, serta pengukuran parameter kinerja seperti waktu tunggu *waiting time*, waktu *turn around*, dan jumlah *context switching*. Hasil simulasi menunjukkan bahwa FCFS lebih sederhana namun menyebabkan waktu tunggu lebih tinggi hasil nyata dari FCFS rata-rata *waiting time* (WT) dan hasilnya 13.1 sedangkan RR memberikan distribusi eksekusi yang lebih adil tetapi meningkatkan jumlah *context switching* RR rata-rata *Waiting time* (WT) hasilnya 17.4 quantum 3. Kesimpulannya, algoritma RR lebih efektif dalam sistem yang memerlukan keadilan dan responsivitas tinggi, sedangkan FCFS lebih sesuai untuk beban kerja ringan tanpa kebutuhan interaktif tinggi.

Kata kunci: Algoritma, Penjadwalan, FCSFS, RR.

Pendahuluan

Manajemen proses dalam sistem operasi adalah bagian penting yang mengatur bagaimana proses di eksekusi dan sumber daya yang mereka butuhkan dialokasikan. ini mencakup proses penjadwalan, seperti algoritma FCFS dan RR, serta mengelola *Context Switching*, dimana sistem operasi beralih antara proses yang berjalan. di mana sistem operasi menjalankan tugas sebagai penghubung antara perangkat lunak dan perangkat keras yang gunakan oleh user. dalam perkembangannya sistem operasi pada awalnya hanya dapat menjalankan proses tunggal atau sekuensal, berubah menjadi kompleks sehingga dapat menjalankan beberapa proses secara bersamaan multitasking. Proses berkomunikasi satu sama lain melalui mekanisme seperti *Inter Process Communication* (IPC). Selain itu, manajemen proses menangani pembuatan dan penghentian proses, serta memastikan penggunaan efisien dari CPU penjadwalan CPU merupakan hal yang penting dalam multitasking dan multiprocessing sebuah sistem operasi karena banyaknya proses yang perlu di jalankan dalam computer. Ada juga merupakan bagian dari fungsi dari , memori, dan perangkat I/O [1] . Penjadwalan proses adalah mekanisme dalam sistem operasi yang menentukan urutan eksekusi proses oleh CPU, memastikan penggunaan sumber daya secara efisien. dan ada tiga jenis penjadwalan yaitu *long-term* untuk memuat proses ke memori ketika ada upaya untuk mengeksekusi suatu proses, penerimaannya ke dalam rangkaian proses yang sedang dieksekusi akan diotorisasi atau ditunda oleh penjadwal jangka panjang, *short-term* untuk menentukan proses Penjadwal jangka pendek juga dikenal sebagai penjadwal CPU memutuskan proses dimana dalam antrian keadaan siap, dalam memori yang akan dieksekusi dialokasikan CPU berikutnya setelah interupsi *clock*, interupsi *Input-Output* (IO) dan panggilan *Operation System* (OS) atau bentuk sinyal lainnya dan *medium-term* untuk penjadwal jangka menengah Penjadwal jangka menengah untuk sementara menghapus proses dari memori utama dan menempatkannya pada memori sekunder seperti *Disk Drive* atau sebaliknya [2], [3] .

Masalah yang sering muncul di penjadwalan proses dalam sistem operasi sering menghadapi berbagai masalah yang dapat mempengaruhi kinerja dan efisiensi sistem. Salah satu masalah utama adalah *starvation*, di mana proses dengan prioritas rendah tidak pernah mendapatkan eksekusi karena terus-menerus diabaikan oleh penjadwal karena tidak ada kesempatan untuk melayani *starvation* pada proses dapat dihindari, karena setiap proses mendapatkan giliran untuk dieksekusi oleh CPU [4]. *Deadlock* juga menjadi isu serius, terutama ketika proses saling menunggu sumber daya yang dipegang oleh proses lain, menghentikan semua eksekusi Jika terlalu besar, algoritma ini akan sama saja dengan algoritma FCFS. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang [5], [6] .selain itu, *Convoy Effect* terjadi ketika proses dengan waktu eksekusi panjang menahan antrian, menyebabkan proses lain mengalami penundaan yang signifikan [7].

terutama jika ukuran *time quantum* Jika terlalu besar, algoritma ini akan sama saja dengan algoritma FCFS. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang pada dasarnya algoritma ini sama dengan FCFS, hanya saja bersifat preemptive. Setiap proses mendapatkan waktu CPU yang disebut dengan waktu kuantum (*quantum time*) untuk membatasi waktu proses [8] .

Throughput yang baik juga merupakan tantangan, Karena algoritma harus memastikan CPU digunakan secara efisien tanpa mengorbankan jumlah proses yang selesai. *Throughput* memaksimalkan jumlah pekerjaan yang diproses per jam. jumlah kerja yang dapat diselesaikan dalam satu unit waktu [9] .terakhir, menjaga adil *Fairnest* dalam penjadwalan adalah kunci, karena algoritma harus memberikan waktu eksekusi yang adil *Fairnest* untuk semua proses, terlepas dari prioritasnya, untuk memastikan sistem berfungsi secara optimal. proses-proses yang diperlakukan sama, dan mendapat jatah waktu pemroses yang sama dan tak ada proses yang tak kebagian layanan pemroses sehingga mengalami kekurangan waktu [10] .FCFS adalah algoritma penjadwalan yang menjalankan proses sesuai urutan kedatangan mereka di antrian, menggunakan metode *First-In, First-Out* (FIFO). Algoritma ini bersifat non-preemptive, artinya begitu sebuah proses mendapatkan CPU, proses tersebut akan berjalan hingga selesai sebelum proses berikutnya dieksekusi. kelebihan utama FCFS adalah kesederhanaannya dan kemudahan implementasi, serta memastikan tidak ada proses yang mengalami *Starvation*. namun kelemahannya termasuk *Convoy Effect*, di mana proses dengan waktu eksekusi panjang dapat menunda proses lain, dan waktu respons yang buruk, terutama dalam sistem interaktif [11]. FCFS cocok untuk sistem dengan beban kerja yang ringan atau proses dengan burst time seragam, tetapi mungkin tidak optimal untuk sistem yang memerlukan respons cepat atau memiliki variasi burst time yang besar. bisa diartikan sebagai Proses yg tiba lebih dahulu akan dilayani lebih dahulu Jika ada proses tiba pada waktu yg sama, maka pelayanan mereka dilaksanakan melalui urutan mereka dalam antrian. Proses di antrian belakang harus menunggu sampai semua proses di depannya selesai Setiap proses yang berada pada status *Ready* dimasukkan ke dalam FCFS *Queue* adalah strutur data antrian, sesuai dengan waktu kedatangannya [12], [13].

RR adalah algoritma penjadwalan yang sering digunakan dalam sistem operasi, khususnya untuk lingkungan *Time-Sharing*. Metode pada penjadwalan algoritma ini membagi waktu CPU secara merata dan membagi semua proses dengan porsi yang sama penjadwalan ini cukup adil karena tidak ada antrian yang diprioritaskan,semua mendapat jatah waktu yang sama. [14], [15] . di antara semua proses dalam antrian dengan menggunakan *Quantum*, yaitu jatah waktu eksekusi yang sama untuk setiap proses. Setelah *Time Quantum* habis, CPU berpindah ke proses berikutnya dalam antrian, menjadikan RR bersifat *preemptive*. tidak ada prioritas dalam proses dan diberikan *Quantum* atau *Time-Slice* yang sama. Penjadwalan *Preemptive* proses diberi jatah waktu oleh pemroses, maka pemroses dapat diambil alih proses lain, sehingga proses disela sebelum selesai dan harus dilanjutkan menunggu jatah waktu pemroses tiba kembali pada proses itu Ketentuannya adalah jika kwanta habis dan proses belum selesai, maka pemroses dialihkan ke proses lain dan jika kwanta belum habis tapi proses telah selesai, maka proses diakhiri dan pemroses dialihkan ke proses lain [16] . RR dibuat sehingga memenuhi kebutuhan *Time-Sharing*. kelebihan utama RR adalah keadilan dan responsivitas, memastikan semua proses mendapatkan kesempatan yang sama untuk dijalankan. penjadwalan proses yang efisien seperti RR dan *Priority* dalam beberapa algoritma penjadwalan, seperti algoritma penjadwalan prioritas, setiap proses memiliki prioritas tertentu [17]. prioritas ini menentukan urutan eksekusi proses, di mana proses dengan prioritas lebih tinggi akan dieksekusi lebih dahulu daripada yang lain. bisa menjadi solusi penting namun, kelemahan utamanya adalah peningkatan *Overhead* dari *Context Switching* yang sering terjadi, terutama jika *Time Quantum* terlalu kecil. pemilihan *Time Quantum* yang tepat sangat penting untuk menjaga efisiensi algoritma ini [18] .

RR *Scheduling* adalah versi preemptive dari FCFS *Scheduling*. Pada RR proses dimasukkan pada *First In First Out Sequence* tetapi setiap proses diijinkan untuk berjalan dalam waktu yang terbatas. Interval waktu ini biasa disebut *Time Slice* atau *Quantum*. dan penjadwalan proses menerapkan strategi *Preemptive,* bukan di-*Preempt* oleh proses lain, tapi terutama oleh penjadwal berdasarkan jatah waktu pemroses yang disebut kwanta (*Quantum*). Jika suatu proses tidak selesai dieksekusi hingga *Time Slice* habis, maka proses tersebut preempted atau berhenti hingga mendapat giliran dieksekusi kembali [19] . algoritma RR merupakan algoritma yang umum digunakan dalam aplikasi waktu nyata *Real-Time*. Peralihan konteks *Context Switching* merupakan salah satu konsep penting dalam algoritma penjadwalan RR. Algoritma RR *Pre-Emption* sistem memberikan

alokasi potongan peralihan konteks. Jika peralihan konteks selesai, proses saat ini akan di*preemptive*, dan akan ditempatkan di belakang antrean siap. Penerapannya dalam sistem waktu nyata *real-time* dan sistem tertanam *embedded.* namun, algoritma RR klasik memiliki waktu penyelesaian yang tinggi, *Throughput* yang kecil, waktu tunggu yang tinggi, dan peralihan konteks yang tinggi [20] . Penjadwalan *Real-Time* pada dasarnya untuk menentukan urutan di mana berbagai *Task* yang harus diambil untuk dieksekusi atau dijalankan oleh sistem operasi. Setiap sistem operasi menggunakan satu atau lebih penjadwalan *Task* untuk mempersiapkan jadwal eksekusi berbagai *Task* yang dibutuhkan untuk dijalankan. Setiap penjadwalan *Task* ditandai oleh algoritma penjadwalan yang mengaturnya. penjadwalan *Real-Time* harus menghasilkan respon yang tepat dalam batas waktu yang telah ditentukan. Jika respon komputer melewati batas waktu tersebut, maka terjadi degradasi performansi atau kegagalan sistem [21] . algoritma RR dapat juga disebut sebagai *Fair Time Scheduling*, yang dimana menggunakan *Quantum Time* karena sumber dari seluruh antrian sama. salah satu algoritma penjadwalan proses paling tua yang mudah diimplementasikan karena kesederhanaannya. karena penjadwalan proses ini tidak menggunakan prioritas maka semua proses dianggap mempunyai kepentingan yang sama sehingga diberikan waktu yang bernama *Time Quantum* atau *Time Slice* [22].

Kecuali penjadwalan RR, sebagian besar algoritma ini dianggap tidak efektif dalam sistem operasi *Real-Time* karena kinerjanya yang buruk. namun, pada sistem operasi *Time-Sharing* dan sistem *Real-Time,* RR berkinerja lebih baik. Algoritma RR telah diuji menggunakan sistem operasi *Real-Time*, yaitu RTOS, dan ditemukan bahwa sistem ini akan bekerja dengan baik karena dikonfigurasi dengan benar untuk menangani proses penjadwalan secara *Real-Time*. algoritma ini menggunakan *Time Slice* tetap. Semua penelitian sebelumnya yang berbasis RR memodifikasi metode *Time Slice*. namun, pendekatan yang berbeda menunjukkan keterbatasan yang berbeda. ketika time slice terlalu besar, proses dalam antrean siap menjadi kekurangan [23] . algoritma Penjadwalan *Weighted Round Robin* (RR) dalam mengatasi beban *Webserver* dimana pada penelitian ini algoritma penjadwalan *Weighted Round Robin* (RR) di implementasikan dalam hal mendistribusikan beban pada *Webserver*, dan pada akhir penelitian didapati bahwa penggunaan algoritma tersebut dapat meningkatkan hasil dari *Response Time Server* dan *Throughput* yang sangat signifikan [24] . Ada juga contoh dari aplikasi pemesanan tiket Bus BMA Trans Makassar berbasis web memiliki kesamaan konsep dengan penelitian perbandingan algoritma penjadwalan FCFS dan RR, yaitu sama-sama berfokus pada efisiensi waktu dan pemerataan layanan. Seperti halnya penjadwalan proses di sistem operasi, sistem ini memerlukan pengaturan urutan pemrosesan data pemesanan, pengecekan kursi, dan jadwal keberangkatan agar dapat diakses cepat dan adil oleh banyak pengguna. Penerapan metode penjadwalan yang tepat akan mempercepat respon, mengurangi waktu tunggu, dan meningkatkan kepuasan pengguna [25] .

Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja algoritma penjadwalan FCFS dan RR dalam mengelola proses pada sistem operasi, khususnya dari aspek rata-rata waktu tunggu dan waktu turnaround. melalui simulasi 10 proses dengan parameter yang telah ditentukan, diperoleh gambaran bahwa setiap algoritma memiliki keunggulan dan keterbatasan masing-masing, di mana FCFS unggul dalam kesederhanaan implementasi tetapi cenderung menyebabkan waktu tunggu tinggi pada proses yang datang belakangan, sementara RR memberikan distribusi waktu eksekusi yang lebih adil tetapi dapat menimbulkan overhead jika quantum tidak optimal. Harapannya, hasil penelitian ini dapat menjadi acuan bagi pengembang sistem operasi atau peneliti lain dalam memilih algoritma penjadwalan yang sesuai dengan kebutuhan dan karakteristik beban kerja, sehingga dapat meningkatkan efisiensi, keadilan, dan kinerja keseluruhan sistem.

Metode

Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja dua algoritma penjadwalan proses dalam sistem operasi, yaitu FCFS dan RR. metode yang digunakan melibatkan beberapa tahapan penting, seperti pemilihan algoritma yang akan dibandingkan, pengumpulan data melalui simulasi, pengujian hipotesis terkait keunggulan dan kelemahan masing-masing algoritma, serta penggunaan alat dan teknologi yang relevan untuk implementasi dan analisis. Pendekatan ini memastikan evaluasi yang komprehensif dan objektif dalam menentukan efisiensi dan efektivitas kedua algoritma tersebut dalam berbagai skenario operasional. adapun bentuk kerangka kerha penilitan yang akan dilakukan sebagai berikut:

A. Mengidentifikasi Ruang Lingkup Masalah

Mengidentifikasi ruang lingkup dalam penelitian ini bertujuan untuk menentukan fokus utama, yaitu menganalisis dan membandingkan algoritma penjadwalan FCFS dan RR. Ini melibatkan penetapan batasan seperti jenis skenario yang diuji, jumlah proses, dan metrik kinerja yang akan diukur. dengan mendefinisikan ruang lingkup secara tepat, penelitian dapat lebih efisien dan menghasilkan analisis yang relevan serta aplikatif.

B. Analisa Masalah

Penelitian ini mengidentifikasi tantangan utama dari algoritma penjadwalan FCFS dan RR. Pada FCFS, masalah *Convoy Effect* terjadi ketika proses dengan waktu eksekusi panjang menyebabkan penundaan proses lain. Sementara itu, pada RR, *Overhead* dapat meningkat akibat seringnya *Context Switching*, terutama jika *Time Quantum* terlalu singkat. resiko *Starvation* proses dengan prioritas rendah tidak terlayani dan *Deadlock* proses saling menunggu sumber daya juga dievaluasi, karena keduanya dapat setiap algoritma dapat memengaruhi kinerja dan efisiensi sistem operasi.

C. Menentukan Tujuan

Menentukan tujuan penelitian adalah langkah penting untuk mengklarifikasi apa yang ingin di capai melalui penelitian tersebut. tujuan penelitian harus jelas,terukur, dan sesuai dengan permasalahan yang di identifikasi. dalam konteks penelitian yang anda sebutkan, tujuan berfokus pada pencapaian akurasi dalam analisis perbandingan algoritna penjadwalan FCFS dan RR pada penjadwalan proses tujuan utama adalah memberikan rekomendasi mengenai algoritma yang lebih optimal dalam situasi tertentu, guna meningkatkan kinerja dan keadilan dalam penjadwalan proses.

D. Mempelajari Literatur Yang Berkaitan Dengan Judul

Dengan mempelajari literatur yang relevan, peneliti dapat mengidentifikasi celah dalam penelitian sebelumnya dan merancang penelitian yang berkontribusi baru serta lebih mendalam, menjadikan penelitian ini tidak hanya relevan, tetapi juga aplikatif dalam mengatasi masalah-masalah yang belum terpecahkan.

E. Mengumpulkan Data-Data Yang Dibutuhkan

Proses pengumpulan data adalah tahap krusial dalam penelitian ini, di mana data yang relevan diperlukan untuk menganalisis dan membandingkan kinerja algoritma penjadwalan FCFS dan RR. data yang dibutuhkan mencakup informasi tentang berbagai skenario proses, seperti jumlah proses, durasi *Burst Time*, dan frekuensi *Context Switching*. untuk memastikan hasil yang akurat, data akan diperoleh melalui simulasi komputer, di mana berbagai skenario operasional diimplementasikan untuk kedua algoritma tersebut. Selain itu, data historis dari literatur yang telah dipelajari juga dapat digunakan sebagai referensi atau pembanding dalam menganalisis hasil simulasi. Parameter penting seperti waktu eksekusi, *Throughput*, dan jumlah *Context Switching* akan diukur dan dicatat dengan cermat selama simulasi. Hasil pengukuran ini akan dianalisis untuk mengevaluasi efisiensi dan efektivitas masing-masing algoritma dalam berbagai kondisi, guna memastikan bahwa penelitian ini didasarkan pada data yang valid dan relevan.

F. Menganalisa Data-Data Yang Telah Ada

Konsep analisis perbandingan algoritma penjadwalan FCFS dan RR dalam sistem operasi merupakan hasil dari perkembangan kolektif dalam ilmu komputer dan sistem operasi. FCFS adalah salah satu metode penjadwalan yang paling awal dan sederhana, muncul bersamaan dengan pengembangan awal sistem operasi batch di mana proses diproses sesuai urutan kedatangan mereka. karena kesederhanaannya, FCFS tidak dapat dikaitkan dengan penemu tertentu. Sebaliknya, algoritma RR dikembangkan pada tahun 1960-an dan 1970-an untuk memenuhi kebutuhan lingkungan *Time-Sharing*, di mana setiap proses diberikan giliran eksekusi secara adil dalam waktu terbatas. Penelitian ini menganalisis data yang telah dikumpulkan untuk mengevaluasi kelebihan dan kelemahan dari kedua algoritma dalam berbagai skenario operasional. FCFS cenderung lebih efisien dalam kondisi beban kerja seragam tetapi rentan terhadap C*onvoy Effect*, di mana proses dengan *Burst Time* panjang memperlambat proses lainnya. sebaliknya, RR lebih unggul dalam situasi yang memerlukan responsivitas tinggi dan eksekusi adil, meskipun efektivitasnya sangat bergantung pada pengaturan *Time Quantum* yang tepat untuk menghindari peningkatan *Context Switching*. Studi ini memberikan wawasan tentang bagaimana masing-masing algoritma dapat dioptimalkan dalam

berbagai kondisi sistem operasi dan memberikan rekomendasi untuk penggunaan algoritma yang paling sesuai berdasarkan kebutuhan operasional.

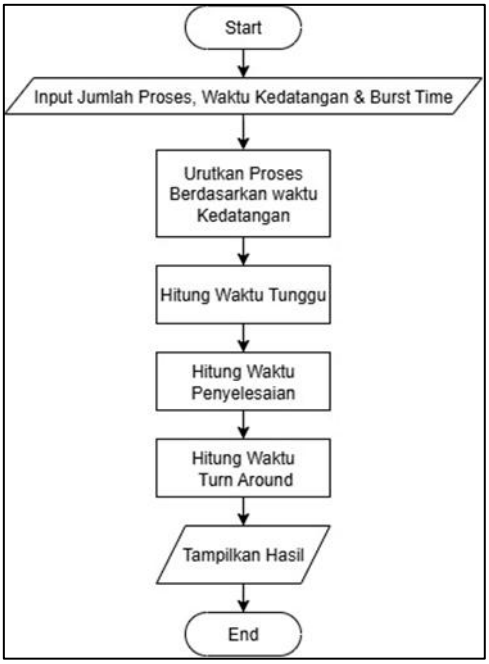
Didalam melakukan perancangan analisis perbandingan algoritma penjadwalan FCFS dan RR pada penjadwalan proses ada berapa tahapan yang harus di lalui

1. Identifikasi tujuan dan ruang lingkup Menetapkan tujuan penelitian, yaitu membandingkan kinerja FCFS dan RR, serta menentukan skenario, jumlah proses, dan metrik kinerja yang akan diukur.
2. Pengembangan simulasi mendesain model simulasi untuk menguji kedua algoritma dengan berbagai skenario beban kerja dan parameter sistem.
3. Pengumpulan data menjalankan simulasi untuk mengumpulkan data mengenai waktu eksekusi, *Context Switching*, dan *Throughput*.
4. Analisis data menganalisis hasil simulasi untuk mengevaluasi kelebihan dan kekurangan masing-masing algoritma.
5. Evaluasi dan interpretasi menilai hasil untuk merekomendasikan algoritma penjadwalan yang optimal berdasarkan kondisi operasional.
6. Pelaporan dan dokumentasi menyusun laporan penelitian yang mencakup temuan, perbandingan, dan rekomendasi.

Perancangan

A. *Flowchart*

Flowchart algoritma penjadwalan FCFS dimulai dengan memasukkan data proses, termasuk waktu kedatangan dan waktu eksekusi. Selanjutnya, dimana proses diurutkan berdasarkan waktu kedatangan untuk memastikan eksekusi dilakukan sesuai urutan masuknya. Jika proses pertama dalam antrian kemudian dieksekusi, dan sistem memeriksa apakah proses tersebut telah selesai dan jika sudah selesai, sistem melanjutkan ke proses berikutnya. jika belum, eksekusi dilanjutkan hingga sampai selesai. Setelah satu proses selesai, sistem memeriksa apakah masih ada proses lain dalam antrian. Jika masih ada, proses berikutnya dieksekusi dengan mekanisme yang sama, dan jika tidak, sistem menampilkan hasil berupa waktu penyelesaian, waktu tunggu, dan waktu putar balik dari semua proses. akhirnya, algoritma berakhir setelah semua proses selesai dieksekusi. Gambar 1 merupakan *flowchart* dari algoritma penjadwalan FCFS.



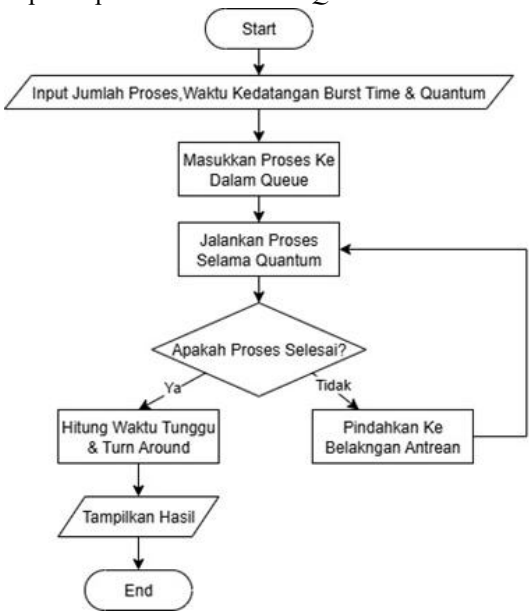
Gambar 1. *Flowchart* cara kerja *First Come First Served* (FCFS)

1. Mulai (*Start*)
2. Proses penjadwalan dimulai dengan menginisialisasi daftar proses yang akan dieksekusi.

3. Input Jumlah Dan Detail Proses
   pengguna memasukkan jumlah proses yang akan dijadwalkan. untuk setiap proses, pengguna memberikan *Burst Time* (BT) dan *Arrival Time* (AT).
4. Sorting Berdasarkan *Arrival Time* (AT)
   Semua proses diurutkan berdasarkan waktu kedatangan *Arrival Time* (AT). jika ada beberapa proses dengan waktu kedatangan yang sama, mereka diproses sesuai urutan input.
5. Inisialisasi Eksekusi Proses
   waktu saat ini *Current Time* dimulai dari waktu kedatangan proses pertama. proses pertama yang masuk ke dalam antrian akan langsung dieksekusi oleh CPU.
6. Eksekusi Proses Secara Berurutan
   Proses pertama dieksekusi hingga selesai tanpa interupsi. setelah proses pertama selesai, proses berikutnya diambil dari antrian dan dijalankan hingga selesai. langkah ini diulang hingga semua proses telah dieksekusi.
7. Hitung Metrik Kinerja
   Waiting Time (WT) dihitung sebagai selisih antara waktu mulai eksekusi dan waktu kedatangan. *Turn Around Time* (TAT) dihitung sebagai total waktu yang dihabiskan proses sejak kedatangannya hingga selesai dieksekusi. *Response Time* (RT) dihitung sebagai waktu pertama kali proses dieksekusi setelah tiba.
8. Tampilkan Hasil
   Setelah semua proses selesai, sistem menampilkan *Waiting Time*, *Turn around Time*, dan *Response Time* untuk setiap proses. rata-rata dari metrik ini dihitung untuk mengevaluasi kinerja algoritma.
9. Selesai *(End)*
   Penjadwalan selesai setelah semua proses dieksekusi.

B. Algoritma *Round Robin*

Algoritma penjadwalan RR dimulai dengan memasukkan jumlah proses, *Burst Time*, dan *Time Quantum*. Setelah itu, proses-proses dimasukkan ke dalam antrean *Ready Queue*. CPU kemudian menjalankan proses pertama selama durasi *Time Quantum* yang telah ditentukan. jika proses belum selesai setelah *Time Quantum* habis, maka proses tersebut dikembalikan ke antrean dan CPU beralih ke proses berikutnya. Jika proses telah selesai, maka sistem mencatat waktu penyelesaian dan melanjutkan ke proses berikutnya. Siklus ini terus berulang hingga semua proses selesai dieksekusi. *Flowchart* pada Gambar 2 menggambarkan bagaimana *Round Robin* memastikan bahwa setiap proses mendapat jatah waktu eksekusi yang adil melalui mekanisme preemption berbasis *Time Quantum*.



Gambar 2. *Flowchart* cara kerja *Round Robin*

1. Mulai *(Start)*

Penjadwalan dimulai.

2. Input Data Proses
   Pengguna memasukkan jumlah proses, *Arrival Time* (AT), *Burst Time* (BT), dan *Time Quantum* (TQ).
3. Urutkan Proses Berdasarkan *Arrival Time* (AT)
   Proses yang tiba lebih awal dimasukkan ke antrian lebih dulu.
4. Inisialisasi Waktu & Jalankan Proses –
   CPU mengeksekusi proses pertama selama maksimal *Time Quantum* (TQ). jika proses selesai sebelum time quantum habis, lanjut ke proses berikutnya. Jika tidak selesai, proses dikembalikan ke antrian untuk dieksekusi lagi pada giliran berikutnya.
5. Periksa Status Proses jika *Burst Time* (BT) > *Time Quantum* (TQ), proses akan dipindahkan ke akhir antrian dan waktu eksekusi dikurangi sebesar TQ. Jika *Burst Time* (BT) ≤ TQ, proses akan selesai dan keluar dari antrian.
6. Periksa Antrian Proses jika masih ada proses dalam antrian, ulangi eksekusi dari langkah 4. Jika semua proses selesai, lanjut ke langkah berikutnya.
7. Hitung Metrik Kinerja *Waiting Time* (WT) = Total waktu tunggu setiap proses dalam antrian sebelum mendapatkan giliran eksekusi. *Turnaround Time* (TAT) = Waktu total dari kedatangan hingga penyelesaian proses. *Response Time* (RT) = Waktu pertama kali proses mulai dieksekusi setelah tiba di sistem.
8. Tampilkan Hasil
   Semua metrik kinerja dan urutan eksekusi proses ditampilkan.
9. Selesai *(End)*
   Semua proses telah selesai dieksekusi.

Pemodelan

A. Tabel hasil perhitungan algoritma penjadwalan FCS

Tabel 1. Perhitungan Proses FCFS

Proses	AT (Arrival Time)	BT (Burst Time)	ST (Start Time)	CT (Completion Time)	TAT (Turnaround Time)	WT (Waiting Time)	RT (Response Time)
P1	0	5	0	5	5	0	0
P2	1	3	5	8	7	4	4
P3	2	8	8	16	14	6	6
P4	3	6	16	22	19	13	13
P5	4	2	22	24	20	18	18
P6	5	4	24	28	23	19	19
P7	6	7	28	35	29	22	22
P8	7	1	35	36	29	28	28
P9	8	3	36	39	31	28	28
P10	9	2	39	41	32	30	30

Pada Tabel 1 hasil perhitungan FCFS menyajikan informasi penjadwalan proses berdasarkan urutan kedatangan (*Arrival Time*) tanpa interupsi. Proses yang datang lebih dulu akan dijalankan terlebih dahulu hingga selesai.

Keterangan:

1. AT *(Arrival Time)*: Waktu proses tiba.
2. BT *(Burst Time)*: Lama proses membutuhkan CPU.
3. ST (*Start Time*): Waktu proses mulai dijalankan.
4. CT (*Completion Time*): Waktu proses selesai.
5. TAT (*Turnaround Time*) = CT - AT
6. WT (*Waiting Time*) = TAT – BT
7. RT (*Response Time*) = WT (Karena FCFS *Non-Preemptive*

B. Tabel perhitungan algoritma penjadwalan RR bergilir dengan *Quantum*. Tabel 2 membantu menilai efisiensi algoritma melalui rata-rata waktu tunggu dan *turn around*

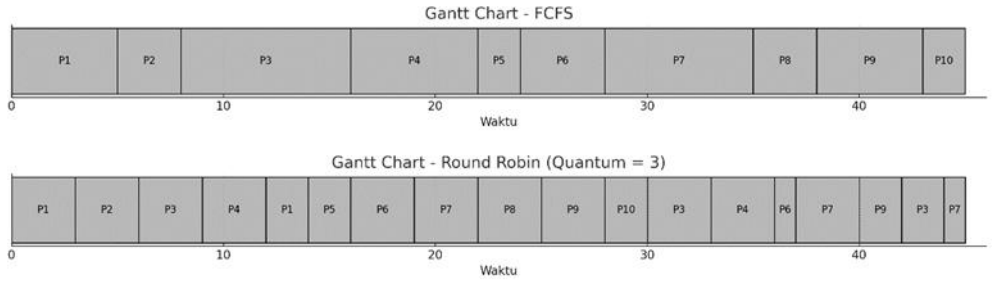
Tabel 2. Perhitungan proses RR

Proses	AT	BT	CT	TAT = CT – AT	WT = TAT - BT	RT (Waktu pertama kali dijalankan - AT)
P1	0	5	33	33	28	0
P2	1	3	18	17	14	4
P3	2	8	39	37	29	8
P4	3	6	36	33	27	12
P5	4	2	16	12	10	16
P6	5	4	29	24	20	20
P7	6	7	41	35	28	24
P8	7	1	14	7	6	28
P9	8	3	25	17	14	28
P10	9	2	21	12	10	28

Proses dieksekusi secara bergilir selama maksimal 4 unit waktu hingga selesai. Proses yang belum selesai dikembalikan ke antrean.

Keterangan:

1. AT: Waktu kedatangan proses.
2. BT: *Burst Time* (durasi CPU).
3. CT: *Completion Time* (waktu proses selesai).
4. TAT *(Turnaround Time)* = CT - AT.
5. WT *(Waiting Time)* = TAT - BT.
6. RT *(Response Time)* = waktu saat pertama kali dijalankan – AT



Gambar 4. *Gant Chart*

Gantt Chart daftar potongan eksekusi *Time Slices* untuk RR : (*Process, Start Time, End Time*).

Tabel 3. Perhitungan Excel FCFS

Process	Arrival Time	Burst Time	Start Time	Completion Time	Turnaround Time	Waiting Time
P1	0	5	0	5	5	0
P2	2	3	5	8	6	3
P3	4	8	8	16	12	4
P4	6	6	16	22	16	10
P5	8	2	22	24	16	14
P6	10	4	24	28	18	14
P7	12	7	28	35	23	16
P8	14	5	35	40	26	21
P9	16	3	40	43	27	24
P10	18	6	43	49	31	25

Pada Tabel 3 per-proses *Process, Arrival Time, Burst Time, Start Time, Completion Time, Turnaround Time, Waiting Time. Arrival Times* berjarak (0,2,4,6,8,10,12,14,16,18), sehingga FCFS berjalan hampir tanpa *idle* dan tiap proses banyak yang dapat langsung dilanjutkan sehingga *Completion Time* (CT) relatif padat.

Keterangan

1. *Process* : label proses (P1.P2.P3.P4.P5.P6.P7.P8.P9.P10).

2. *Arrival Time* (AT) : waktu proses tiba di sistem.
3. *Burst Time* (BT) : total CPU time yang dibutuhkan proses.
4. *Start Time* (ST) FCFS : waktu proses pertama kali mulai dieksekusi. untuk proses pertama (ST) = AT; untuk proses berikutnya Start Time (ST) = *max* (CT_sebelumnya, AT).
5. *Completion Time* (CT): waktu proses selesai; rumus: (CT) = ST + BT (untuk FCFS). untuk (RR) dihitung ketika remaining *Burst Time* (BT) menjadi 0.
6. *Turnaround Time* (TAT): total waktu dalam sistem; TAT = CT − AT.
7. *Waiting Time* (WT): waktu proses menunggu di *ready queue*; WT = TAT − BT.
   Rata-rata *Waiting Time* FCFS = (0+3+4+10+14+14+16+21+24+25) / 10 = 13.1

Tabel 4. Perhitungan Excel RR

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	5	8	8	3
P2	2	3	6	4	1
P3	4	8	40	36	28
P4	6	6	31	25	19
P5	8	2	16	8	6
P6	10	4	38	28	24
P7	12	7	49	37	30
P8	14	5	45	31	26
P9	16	3	34	18	15
P10	18	6	48	30	24

Pada table 4 per-proses di *Process, Arrival Time, Burst Time, Completion Time, Turnaround Time, Waiting Time* RR bersifat *preemptive* sehingga *Start Time* per giliran dicatat di *Sheet Gantt*). RR *Gantt Chart* daftar potongan eksekusi (*time slices*) untuk RR : *(Process, Start Time, End Time*). *Round Robin* (RR) dengan quantum kecil 3 memecah proses panjang misalnya (P3 BT=8, P7 BT=7) menjadi banyak potongan. tiap potongan menyebabkannya kembali ke antrian dan menunggu giliran lagi *akumulasi waiting time* tinggi untuk proses-proses panjang. RR menambah jumlah *Context Switches* walaupun adil, *Preemption* dapat menaikkan total *Waiting Time* rata-rata apabila banyak proses panjang dan quantum kecil. rata-rata *Waiting Time* RR , *quantum*=3) = (3+1+28+19+6+24+28+26+15+24)/10 = 17.4
Berdasarkan hasil perhitungan, rata-rata waktu tunggu metode FCFS adalah 13,1 satuan waktu, sedangkan metode RR *quantum* = 3 dan menghasilkan rata-rata waktu tunggu 17,4 satuan waktu. maka hal ini menunjukkan bahwa pada dataset ini, FCFS lebih efisien dalam meminimalkan waktu tunggu dibandingkan RR, meskipun RR memiliki keunggulan dalam keadilan eksekusi proses secara bergilir

Kesimpulan

Dari hasil analisis dan simulasi terhadap algoritma penjadwalan FCFS dan RR, disimpulkan bahwa FCFS lebih sederhana dan cocok untuk proses berurutan dengan beban ringan, namun dapat menyebabkan waktu tunggu tinggi. Sementara itu, RR lebih adil dan responsif untuk sistem multitasking karena membagi waktu secara merata, meskipun berisiko menghasilkan O*verhead* tinggi jika quantum terlalu kecil. metode FCFS menghasilkan rata-rata waktu tunggu yang lebih tinggi dibandingkan metode RR dengan *Quantum* 3. hal ini terjadi karena pada FCFS, proses yang datang lebih awal dengan *burst time* panjang akan membuat proses lain menunggu lebih lama *Convoy Effect*. Sebaliknya, RR membagi waktu eksekusi secara bergilir sehingga waktu tunggu rata-rata menjadi lebih rendah dan distribusi waktu antar proses lebih merata.

Dengan demikian, RR lebih unggul dalam mengurangi waktu tunggu pada sistem multitasking, sedangkan FCFS lebih sesuai untuk skenario dengan jumlah proses sedikit dan beban kerja yang relatif sama. Pemilihan algoritma sebaiknya disesuaikan dengan kebutuhan sistem. hasil ini mendukung tujuan penelitian untuk membandingkan efisiensi kedua algoritma dan menjawab masalah penjadwalan proses dalam sistem operasi.Saran dari penelitian ini untuk memperluas cakupan analisis dengan membandingkan lebih banyak algoritma penjadwalan, seperti *Shortest Job First* (SJF), *Priority Scheduling*, dan *Multilevel Feedback Queue*, sehingga diperoleh gambaran kinerja yang lebih menyeluruh. Selain itu, pada metode RR, penting untuk

menguji variasi nilai *Time Quantum* agar dapat menentukan pengaturan yang paling efisien sesuai karakteristik beban kerja. Penelitian lanjutan juga diharapkan menggunakan data dan simulasi yang lebih kompleks dan mendekati kondisi nyata, sehingga hasilnya dapat lebih akurat dan bermanfaat bagi pengembangan sistem operasi di masa depan.

Daftar Pustaka

[1] M. T. D. Putra, H. Hidayat, N. Septian, and T. Afriani, "Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve (FCFS) Dan Round Robin," *Building of Informatics, Technology and Science (BITS)*, vol. 3, no. 3, pp. 207–212, Dec. 2021, doi: 10.47065/bits.v3i3.1047.

[2] M. A. I. P. Paulus V. Daud Boseren, "Analisis Perbandingan Algoritma Penjadwalan CPU A New Improved Round Robin Dan A Dynamic Time Quantum Shortest Job Round Robin," 2016.

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

[4] U. Agustina, A. sa'dyah, and M. A. Yaqin, "Pengaruh Prioritas Dinamis terhadap Starvation dalam Algoritma Penjadwalan CPU," pp. 1–5, Jun. 2025.

[5] T. Dharma Putra and R. Purnomo, "Analisis Algoritma Round Robin pada Penjadwalan CPU," *Jurnal Ilmiah Teknologi Informasi Asia*, vol. 15, no. 2, 2021.

[6] A. Fajar, D. S. Dhika, and R. P. Febriansyah, "Strategi Penanganan Deadlock Yang Efektif Dalam Sistem Operasi Berbasis Windows: Pencegahan Deadlock, Mengidentifikasi Faktor Penyebab, dan Dampak Dari Deadlock," *Sistem dan Teknologi Informasi Indonesia (SINTESIA)*, vol. 2, no. 1, pp. 6–11, 2022.

[7] A. Zulfahrizan, M. Alby Savana HSB, F. Br.Hutagalung, and F. Ramadhani, "Implementasi Library Python Dequeue Pada Antrian Bank Menggunakan Logika First In First Out," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 9, no. 1, pp. 224–228, Dec. 2024, doi: 10.36040/jati.v9i1.12260.

[8] Muh. Y. Taufiq, La Ode Muh., L.M. Fid Aksara, "Analisi Perbandingan Algoritma Penjadwalan Round Robin Dan Shortest Job First Untuk Manajemen Proses Dalam Single Processing," *semanTIK*, vol. 7, no. 1, pp. 1–5, 2021, doi: 10.5281/zenodo.5036494.

[9] Dr. R. B. G. Neetu Goel, "A Comparative Study Of CPU Scheduling Algorithms," 2012.

[10] M. T. D. Putra, H. Hidayat, N. Septian, and T. Afriani, "Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve (FCFS) Dan Round Robin," *Building of Informatics, Technology and Science (BITS)*, vol. 3, no. 3, pp. 207–212, Dec. 2021, doi: 10.47065/bits.v3i3.1047.

[11] M. R. M. Ramadhan, S. Sarwido, and T. Tamrin, "Implementasi Algoritma First Come First Served Pada Sistem E-Booking Lapangan Dihafa Mini Soccer," *Jutisi : Jurnal Ilmiah Teknik Informatika dan Sistem Informasi*, vol. 13, no. 3, Jan. 2025, doi: 10.35889/jutisi.v13i3.2404.

[12] S. N. H. Parinduri, Ikhsan, "Teknik Penjadwalan Prosesor FIFO, SJF Non Preempetive, Round Robin," *Prosiding Seminar Nasional Riset Information Science (SENARIS)*, 2019.

[13] T. M. Tamba and R. R. Fiska, "Penerapan Algoritma First Come First Served Dan Priority Service Pada Aplikasi Pendaftaran Pasien Klinik Dr. Moris," *remik*, vol. 9, no. 3, pp. 975–982, Aug. 2025, doi: 10.33395/remik.v9i3.15122.

[14] G. Ardi Wijaya, "Implementasi Algoritma Round Robin Pada Sistem Penjadwalan Mata Kuliah," 2018.

[15] A. Wijaya and G. Gunawan, "Implementasi Algoritma Round Robin pada Sistem Penjadwalan Mata Kuliah (Studi Kasus: Universitas Muhammadiyah Bengkulu)," *Jurnal Informatika UPGRIS*, vol. 4, no. 1, p. 465400.

[16] M. Santika and S. Hansun, "Implementasi Algoritma Shortest Job First dan Round Robin pada Sistem Penjadwalan Pengiriman Barang," *ULTIMATICS*, vol. VI, no. 2, 2014.

[17] M. Darip, N. Supiana, and S. Makin, "Penggunaan Algoritma Round Robin Dalam Manajemen Kemitraan dan Reservasi Kendaraan Bagi Wisatawan di Provinsi Banten," *IJIS - Indonesian Journal On Information System*, vol. 9, no. 2, p. 218, Sep. 2024, doi: 10.36549/ijis.v9i2.322.

[18] A. D. Tri Wahyu Prasetyo, Wiharto, "Pemodelan Penjadwalan Multilevel Feedback Queue Menggunakan Dynamic Time Quantum Pada Kasus Pemesanan Makanan Di Restoran," vol. 4, no. 2, 2015.

[19] D. Sofiansyah Fadli, Wire Bagye, S.Kom., M.Kom, Resad Setyadi., S.T., S.Si., MMSI., Ph and M. K. Yesaya Tommy Paulus, S.Kom., MT., Ph.D, Lalu Mutawalli, S.Kom., M.I.Kom., M.Kom, Saruni Dwiasnati, ST., MM., M.Kom, Ida Bagus Ary Indra Iswara, S.Kom., M.Kom, Erlin Windia Ambarsari, Fachrudin Pakaja, S.Kom, M.T, Ahmad Jufri, S.Kom., M.T, Mohammad Taufan, "Sistem Penjadwalan Event Organized Dengan Metode Round Robin (RR)," *MISI (Jurnal Manajemen informatika & Sistem Informasi)*, vol. 3, No 2, 2020.

[20] R. P. Tri Dharma Putra, "Simulation Of Priority Round-Robin Scheduling Algorithm," *Sinkron*, vol. 7, no. 4, pp. 2170–2181, Oct. 2022, doi: 10.33395/sinkron.v7i4.11665.
[21] Ph. D. Dhanny Rukmana Manday, Achmad Imam Kistijantoro, ST, M.Sc, "Analisis Sistem Penjadwalan Real-Time Multicore Pada Virtualisasi Rt-Xen 2.0," 2019.
[22] K. J. A. Wisnu Widiarto, Desinta Maheswari, Dewi Puspita Sari, "Implementasi Algoritma Round Robin Dan Priority Pada Sistem Antrian Rumah Sakit," *Fasilom*, vol. 14 No.2, 2024.
[23] K. A. E. Nermeen Ghazy, Afaf Abdelkader, Mervat S. Zaki, "A New Round Robin Algorithm For Task Scheduling In Real-Time System," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 5, pp. 691–704, Oct. 2022, doi: 10.22266/ijies2022.1031.59.
[24] 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, Feb. 2024, doi: 10.31943/teknokom.v7i1.176.
[25] A. M. Muh Dasriyanto Saleh, Siska Anraeni, "Aplikasi Pemesanan Tiket Bus BMA Trans Makassar Berbasis Web," *Buletin Sistem Informasi dan Teknologi Islam*, vol. 1, no. 1, pp. 51–55, 2020, doi: 10.33096/busiti.v1i1.675.