

Tugas Mata Kuliah  
Sistem Operasi

Simulasi  
*Algoritma Penjadualan Proses*

Firmansyah Adiputra                      Hari Toha Hidayat  
NIM. 10/306872/PPA/3318              NIM. 09/292186/PPA/03058

Program Magister Ilmu Komputer  
Universitas Gadjah Mada  
November 2010

## 1 Pendahuluan

Dalam sistem operasi *multiprogram*, komputer dapat dibuat menjadi lebih produktif dengan jalan melakukan pengalihan kerja CPU di antara proses-proses yang sedang berjalan [1]. Kegiatan pengalihan kerja CPU itulah yang disebut sebagai penjadualan proses atau penjadualan CPU.

Pada tulisan ini akan dibahas aplikasi komputer untuk mensimulasikan beberapa algoritma penjadualan proses. Aplikasi algoritma penjadualan proses yang dibuat oleh penulis diberi nama **PSSAV** (*Process Scheduling Simulation, Analysis, and Visualization*). Beberapa algoritma penjadualan proses yang disimulasikan dalam PSSAV adalah.

- FIRST-COME, FIRST-SERVED *scheduling*;
- SHORTEST-JOB-FIRST *scheduling* (*preemptive* dan *non-preemptive*);
- PRIORITY *scheduling* (*preemptive* dan *non-preemptive*);
- ROUND-ROBIN *scheduling*.

## 2 Algoritma Penjadualan Proses

Pada seksi ini akan dijelaskan beberapa algoritma penjadualan proses yang disimulasikan pada aplikasi PSSAV.

## 2.1 First-Come, First-Served (FCFS) Scheduling

Algoritma penjadualan ini adalah algoritma penjadualan CPU yang paling sederhana. Dalam algoritma ini proses yang pertama kali melakukan *request* CPU akan mendapatkan alokasi CPU yang pertama kali pula. Algoritma FCFS ini dapat diimplementasikan dengan mudah menggunakan antrian FIFO (*First-In First-Out*).

Kelemahan dari algoritma ini adalah rata-rata *waiting time* yang cukup lama. Serta kemungkinan terjadinya *convoy-effect* yaitu proses-proses dalam antrian *ready* akan menunggu dalam waktu lama apabila proses yang sedang dieksekusi membutuhkan waktu yang lama.

## 2.2 Shortest-Job-First (SJF) Scheduling

Dalam algoritma SJF ini, proses yang memiliki CPU *burst* paling kecil akan dialokasikan terlebih dahulu. Bila terdapat lebih dari satu proses dengan CPU *burst* yang sama maka, digunakan algoritma FCFS.

Algoritma SJF dapat dilakukan secara *preemptive* maupun *non-preemptive*. Perbedaan dari kedua jenis algoritma SJF tersebut terjadi saat ada proses dalam antrian *ready* yang memiliki CPU *burst* lebih kecil daripada proses yang sedang dieksekusi. Pada algoritma *preemptive*, proses pada antrian *ready* tersebut akan menggantikan proses yang sedang dieksekusi. Sedangkan pada algoritma *non-preemptive*, proses yang sedang dieksekusi tidak akan digantikan oleh proses lain sampai proses tersebut menyelesaikan CPU *burst*-nya.

Algoritma SJF *preemptive* juga sering disebut sebagai SHORTEST-REMAINING-TIME-FIRST *scheduling*.

## 2.3 Priority Scheduling

Dalam *priority scheduling*, setiap proses memiliki nilai prioritas. Proses dengan prioritas tertinggi akan dialokasikan CPU terlebih dahulu. Jika ada lebih dari satu proses dengan nilai prioritas yang sama maka akan digunakan algoritma FCFS.

Algoritma SJF sebenarnya adalah varian dari *priority scheduling*, dengan penentuan prioritas berdasarkan CPU *burst* dari tiap proses. Proses dengan CPU *burst* lebih besar akan diberi prioritas lebih rendah.

Algoritma *priority scheduling* dapat pula dilakukan secara *preemptive* maupun *non-preemptive*. Perbedaan dari keduanya terjadi saat ada proses dalam antrian *ready* yang memiliki prioritas lebih tinggi daripada proses yang sedang dieksekusi. Pada algoritma *preemptive*, proses pada antrian *ready* tersebut akan menggantikan proses yang sedang dieksekusi. Sedangkan pada algoritma *non-preemptive*, proses yang sedang dieksekusi tidak akan digantikan oleh proses lain sampai proses tersebut menyelesaikan CPU *burst*-nya.

Kelemahan dari *priority scheduling* adalah kemungkinan terjadinya *indefinite blocking* (*starvation*). Sebuah proses dengan prioritas rendah memiliki kemungkinan untuk tidak akan dieksekusi sampai waktu yang tidak bisa ditentukan apabila selalu ada proses

dengan prioritas yang lebih tinggi. Solusi dari permasalahan ini adalah *aging* yaitu meningkatkan prioritas dari setiap proses yang menunggu dalam antrian secara bertahap.

## 2.4 Round-Robin Scheduling

Algoritma round-robin didesain untuk sistem *time-sharing*. Setiap proses akan mendapat jatah eksekusi CPU sebesar nilai *time quantum* (umumnya antara 10 - 100 ms). Jika *time quantum*-nya telah habis atau proses sudah selesai maka, CPU akan dialokasikan ke proses berikutnya. Antrian *ready* pada algoritma ini diperlakukan sebagai antrian *circular*.

Untuk mengimplementasikan *round-robin scheduling*, maka untuk antrian *ready* digunakan antrian FIFO dan proses-proses baru akan ditambahkan ke akhir dari antrian.

Algoritma *round-robin* ini dapat dikatakan cukup adil, jika ada  $n$  proses dalam antrian *ready*, dan *time quantum* adalah sebesar  $q$ . Maka setiap proses akan mendapatkan jatah CPU *time* sebesar  $\frac{1}{n}$  dalam potongan-potongan waktu yang tidak lebih besar dari  $q$ . Setiap proses juga tidak akan menunggu dalam waktu yang lebih lama dari  $(n - 1) \times q$ .

Algoritma ini sepenuhnya bergantung pada besarnya *time quantum*. Pada konfigurasi ekstrem, jika *time quantum* sangatlah besar maka *round-robin* akan berkelakuan seperti FCFS. Jika *time quantum* terlalu kecil, maka sebagian besar proses tidak akan selesai dalam satu *time quantum* dan akan sering dilakukan *context switching*, yang berarti terjadi *overhead* yang semakin besar.

Sebagai acuan, untuk menentukan besarnya *time quantum* adalah *time quantum* harus cukup besar bila dibandingkan dengan waktu yang dibutuhkan untuk melakukan *context switching*. Namun juga jangan terlalu besar, dan sebaiknya 80% dari proses yang ada memiliki CPU *burst* yang lebih kecil daripada *time quantum*.

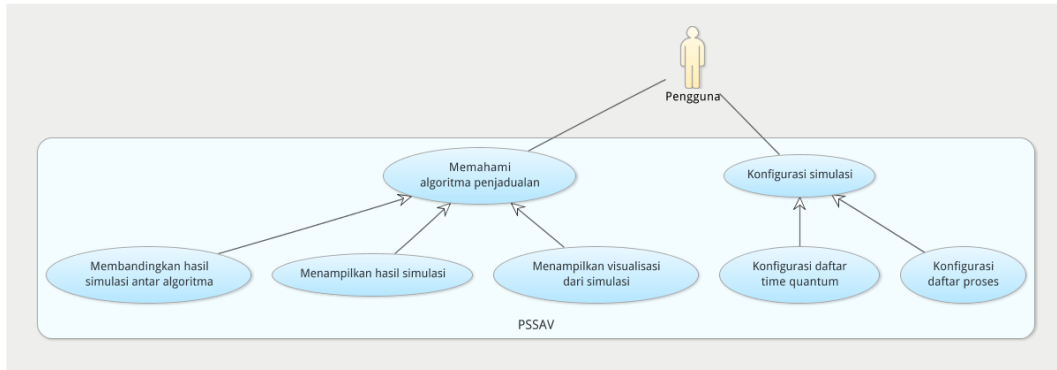
## 3 Aplikasi PSSAV

Aplikasi yang dibuat untuk melakukan simulasi penjadualan proses, pada tulisan ini diberi nama PSSAV (*Process Scheduling Simulation, Analysis, and Visualization*). Pada seksi ini akan dijelaskan secara singkat tentang kebutuhan, perancangan, implementasi, dan ujicoba dari aplikasi PSSAV.

### 3.1 Kebutuhan

Aplikasi yang harus dibuat adalah aplikasi untuk mensimulasikan algoritma penjadualan proses. Berikut adalah beberapa kebutuhan yang harus dipenuhi oleh aplikasi tersebut.

- Input data proses dan batasan-batasan lainnya;
- Dapat memberikan pemahaman pada *user* tentang konsep penjadualan yang disimulasikan.



Gambar 1: *Use case diagram* PSSAV

### 3.2 Perancangan Aplikasi

Berdasarkan dua kebutuhan utama seperti pada seksi 3.1 maka dapat dirancang *use case* seperti pada Gambar 1.

Berikut adalah deskripsi untuk setiap *use cases* yang ada pada Gambar 1.

1. Untuk melakukan konfigurasi terhadap simulasi, ada beberapa hal yang harus dipenuhi oleh aplikasi PSSAV.
  - a) Aplikasi harus memiliki fasilitas untuk memodifikasi konfigurasi proses-proses yang akan disimulasikan yaitu konfigurasi terhadap nilai CPU *burst* (*burst time*), nilai prioritas, dan waktu kedatangan proses (*arrival time*);
  - b) Aplikasi harus memiliki fasilitas untuk memodifikasi konfigurasi nilai-nilai *time quantum* yang akan digunakan pada simulasi algoritma *round-robin*.
2. Untuk membantu pengguna memahami konsep penjadualan proses maka aplikasi PSSAV harus memenuhi kebutuhan berikut.
  - a) Aplikasi harus dapat memberikan gambaran secara visual tentang algoritma penjadualan yang disimulasikan kepada pengguna. Fitur ini dapat berupa penjelasan yang berupa gambar atau animasi. Perlu juga disediakan informasi singkat tentang algoritma penjadualan.
  - b) Aplikasi harus memiliki fasilitas yang dapat membantu pengguna menganalisis data-data yang dihasilkan oleh simulasi penjadualan. Data-data hasil simulasi diantaranya adalah data aktifitas yang terjadi pada setiap CPU *time*, dan profil dari setiap proses.
  - c) Aplikasi harus memiliki fasilitas untuk membandingkan beberapa algoritma penjadualan, dan menampilkan informasi perbandingan tersebut kepada pengguna.

### 3.3 Implementasi Aplikasi

Aplikasi PSSAV ini diimplementasikan dengan menggunakan bahasa pemrograman JAVA dan dibangun di atas NETBEANS PLATFORM 6.9.1.

Berdasarkan *use cases* yang dijelaskan pada seksi 3.2 maka, berikut adalah daftar fitur pada aplikasi PSSAV untuk setiap *use case* yang bersesuaian.

**Use case 1a** pada aplikasi PSSAV, pengguna dapat menambah proses baru, memodifikasinya, dan bahkan menghapusnya, dan semua data simulasi (baik data untuk analisis maupun perbandingan) akan diperbarui secara otomatis.

**Use case 1b** pada aplikasi PSSAV, algoritma *round-robin* bisa memiliki lebih dari satu simulasi dengan nilai *time quantum* yang berbeda. Pengguna dapat menentukan sendiri daftar *time quantum* yang diinginkannya, yang secara otomatis, aplikasi PSSAV, akan membuat simulasi round-robin untuk setiap *time quantum* yang ditentukan, sekaligus memperbarui tampilan dan data-data yang berkaitan dengan simulasi tersebut.

**Use case 2a** pada aplikasi PSSAV, pengguna dapat melihat animasi yang memvisualisasikan kegiatan penjadualan proses. Pengguna juga dapat mengontrol jalannya animasi yang ditampilkan. Animasi tersebut akan diperbarui secara otomatis apabila terjadi perubahan konfigurasi.

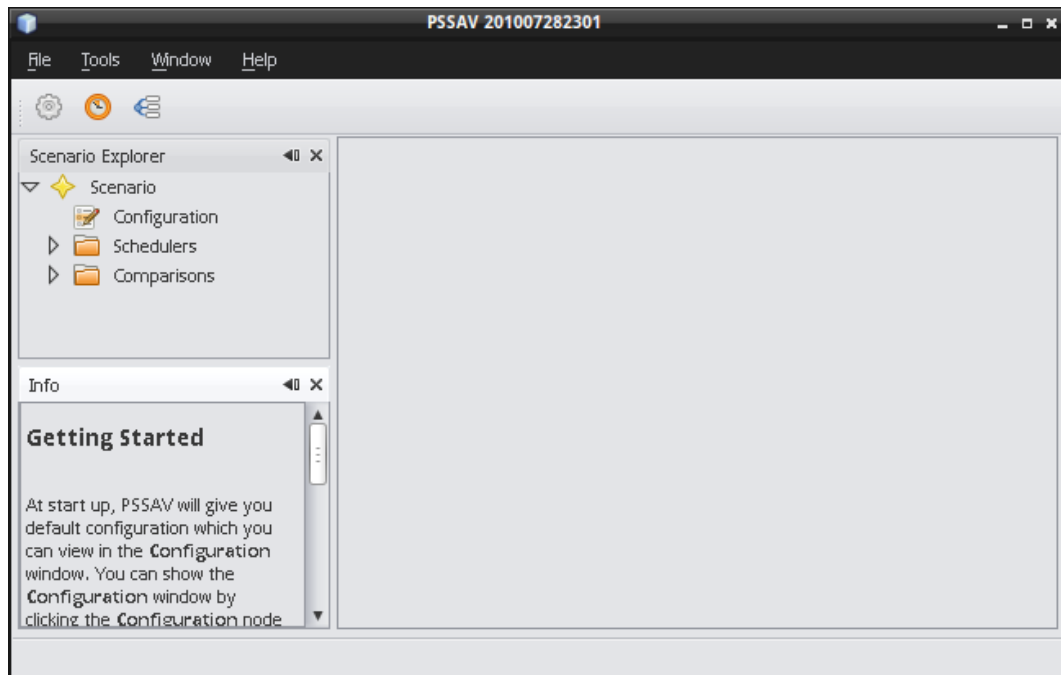
Pada aplikasi PSSAV ini juga terdapat fasilitas *info viewer* yang akan menampilkan informasi-informasi yang berkaitan dengan jendela yang sedang menerima fokus. Saat jendela yang menampilkan data tentang sebuah algoritma mendapatkan fokus, maka akan ditampilkan informasi yang berkaitan dengan algoritma tersebut.

**Use case 2b** pada aplikasi PSSAV, pengguna dapat mengetahui status dari setiap proses pada setiap CPU *time* yang ditampilkan dalam bentuk tabel. Pengguna juga dapat mengetahui profil dari tiap-tiap proses baik dalam bentuk grafik/*chart* maupun dalam bentuk tabular.

**Use case 2c** pada aplikasi PSSAV, pengguna dapat melakukan perbandingan antara dua atau lebih algoritma-algoritma penjadualan proses. Dalam perbandingan tersebut akan ditampilkan perbandingan profil setiap prosesnya (dalam bentuk grafik dan data tabular), perbandingan kinerja dari algoritma berdasarkan rata-rata *waiting time* dan *turnaround time* (dalam bentuk grafik dan data tabular), serta perbandingan secara visual terhadap *Gantt chart* dari tiap algoritma.

### 3.4 Ujicoba Aplikasi

Pada seksi ini akan dijelaskan hasil *running* dari aplikasi PSSAV beserta petunjuk penggunaan aplikasi.



Gambar 2: Tampilan awal PSSAV

### 3.4.1 Tampilan awal




Tampilan awal dari aplikasi PSSAV seperti yang terlihat pada Gambar 2, terdiri dari tiga bagian utama yaitu.

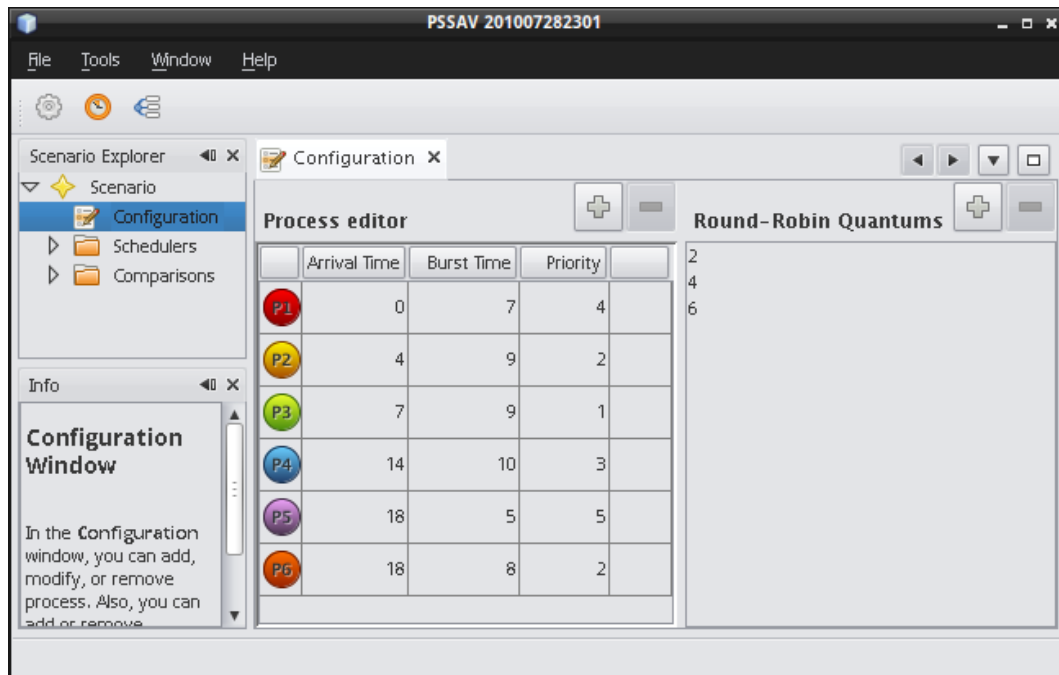
**Scenario Explorer** pada jendela ini ditampilkan *tree view* yang berisi *launcher* untuk menampilkan jendela konfigurasi, data simulasi *scheduler*, dan perbandingan antar *scheduler*.

**Info Viewer** pada jendela ini ditampilkan berbagai informasi dari item yang sedang menerima fokus.

**Editor** area ini adalah area yang akan ditempati oleh jendela konfigurasi, data simulasi *scheduler*, dan perbandingan antar *scheduler*.

Pada *toolbar* terdapat tiga *launcher* yaitu,

-  **New process...** *launcher* ini akan menampilkan *wizard* untuk membuat proses baru.
-  **New round robin quantum...** *launcher* ini akan menampilkan *dialog* untuk menentukan *time quantum* baru yang akan digunakan oleh algoritma *round-robin*.
-  **New comparison...** *launcher* ini akan menampilkan *wizard* untuk membuat perbandingan baru antara dua atau lebih algoritma.



Gambar 3: Jendela Configuration

### 3.4.2 Jendela Konfigurasi

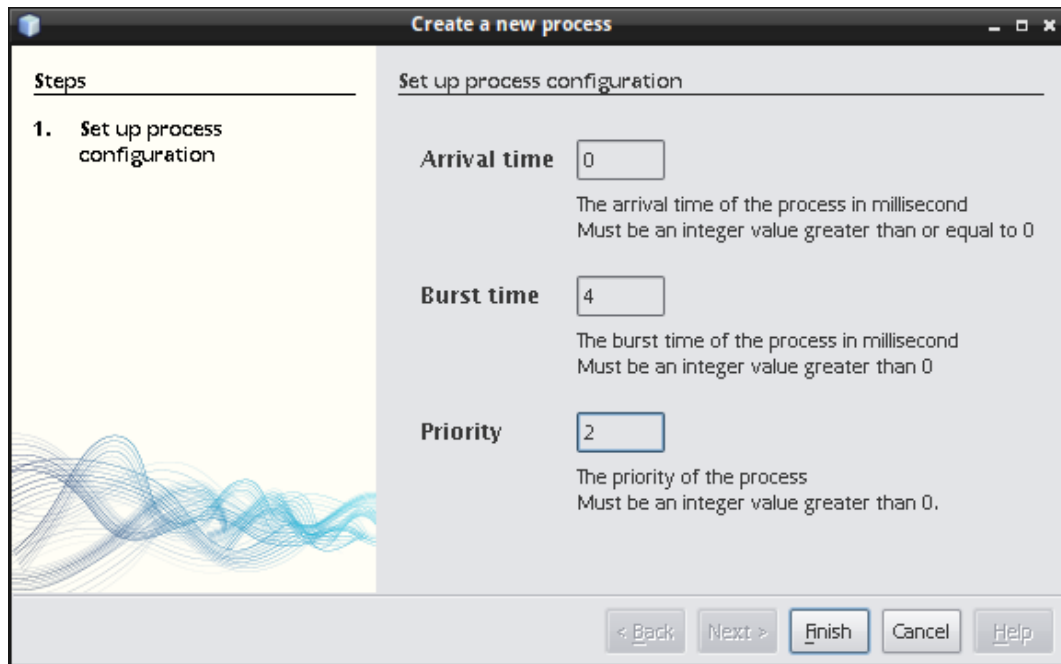
Setiap kali PSSAV dijalankan, secara otomatis PSSAV akan memberikan konfigurasi *default* untuk daftar proses dan daftar *time quantum* (*time quantum* digunakan untuk algoritma *round-robin*). Daftar proses, secara *default*, sama seperti soal UTS Sistem Operasi pada tanggal 25 Oktober 2010 (dengan dosen pengampu Dr. Khabib Mustofa, M.Kom.). Sedangkan daftar *time quantum* secara *default* adalah 2, 4, dan 6.

Konfigurasi *default* tersebut dapat dimodifikasi melalui jendela **Configuration** yang bisa ditampilkan dengan cara menekan launcher **Scenario** > **Configuration**. Tampilan dari jendela **Configuration** bisa dilihat pada Gambar 3.

Daftar proses ditampilkan pada tabel **Process editor**, konfigurasi dari setiap proses dapat dimodifikasi dengan cara memilih nilai yang akan dimodifikasi kemudian mengetikkan nilai yang baru.


Cara penambahan dan penghapusan proses dapat dilihat pada seksi 3.4.3. Sedangkan cara penambahan dan penghapusan nilai *time quantum* dapat dilihat pada seksi 3.4.4.

Setiap perubahan konfigurasi akan secara otomatis berakibat pada data simulasi dan data perbandingan.



Gambar 4: Wizard New process...

### 3.4.3 Konfigurasi Daftar Proses


Proses baru dapat ditambahkan ke daftar proses melalui *wizard New process...* yang dapat ditampilkan dengan cara menekan tombol **New process...** pada toolbar, atau menekan tombol  pada bagian atas tabel Process editor. Tampilan *wizard New process...* dapat dilihat pada Gambar 4.

Pada *wizard New process...*, pengguna diharuskan mengisi data tentang proses yang akan dibuat. Data-data tersebut adalah.

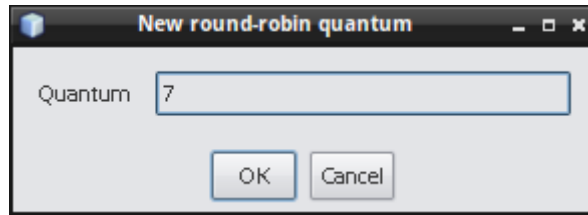
**Arrival time** adalah data waktu simulasi datangnya proses (waktu simulasi pembuatan proses). Data ini harus diisi dengan bilangan bulat yang lebih besar atau sama dengan 0.

**Burst time** adalah data yang menunjukkan CPU *burst* dari proses yang akan dibuat. Data ini harus diisi dengan bilangan bulat yang lebih besar dari 0.

**Priority** adalah data prioritas dari proses yang akan dibuat. Semakin besar nilai prioritas berarti semakin rendah prioritasnya. Data ini harus diisi dengan bilangan bulat yang lebih besar atau sama dengan 1.


Untuk menghapus data proses dari daftar proses, pilih proses yang akan dihapus pada tabel Process editor, kemudian tekan tombol .






Gambar 5: *Wizard New round-robin quantum...*

#### 3.4.4 Konfigurasi Daftar *Round-Robin Time Quantum*

Simulasi *round-robin* untuk *time quantum* tertentu dapat dikonfigurasi dengan jalan menambahkan nilai *time quantum* baru ke daftar *Round-Robin Quantum*s melalui *wizard New round-robin quantum...* yang dapat ditampilkan dengan cara menekan tombol *New round-robin quantum...* pada toolbar, atau menekan tombol  pada bagian atas daftar *Round-Robin Quantum*s. Tampilan *wizard New round-robin quantum...* dapat dilihat pada Gambar 5.

Pada *wizard New round-robin quantum...*, pengguna diharuskan mengisi data tentang *time quantum* baru yang akan dibuat yaitu bilangan bulat yang lebih besar dari 0.

Untuk menghapus data *time quantum* dari daftar *Round-Robin Quantum*s, pilih *time quantum* yang akan dihapus pada daftar *Round-Robin Quantum*s, kemudian tekan tombol .

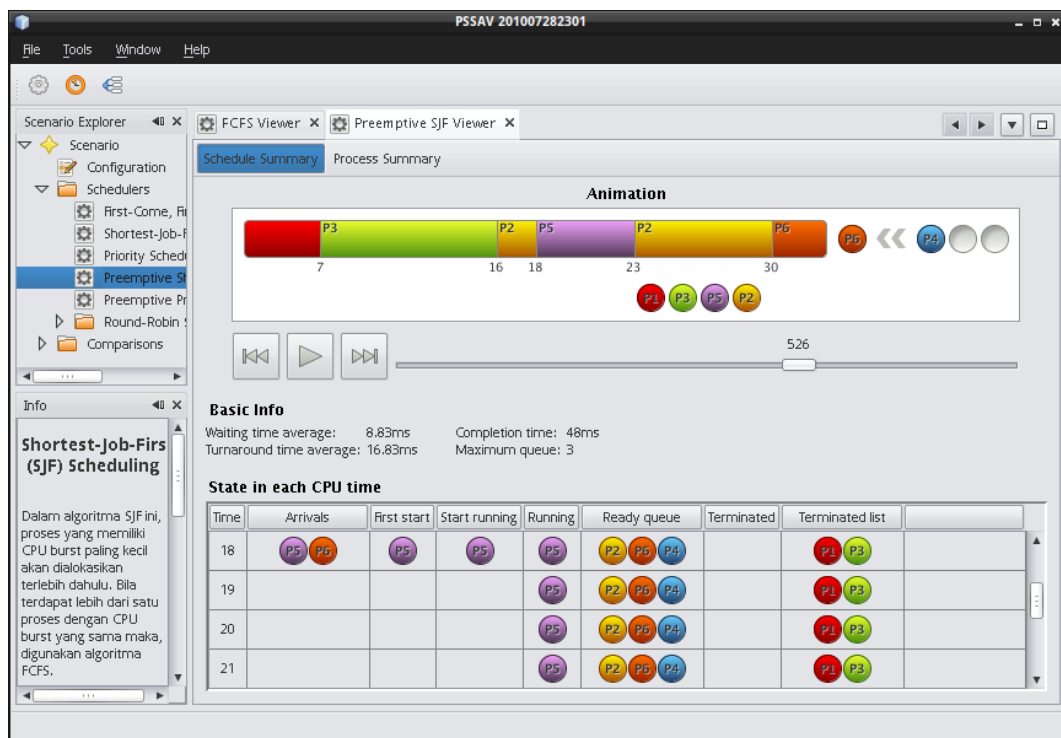
#### 3.4.5 Jendela Data Simulasi

Untuk setiap simulasi data-datanya dapat ditampilkan dengan jalan menekan *sub-node* dari *node Schedulers*. Sebagai contoh, tampilan dari data simulasi *preemptive SJF* akan terlihat seperti pada Gambar 6 dan Gambar 7.

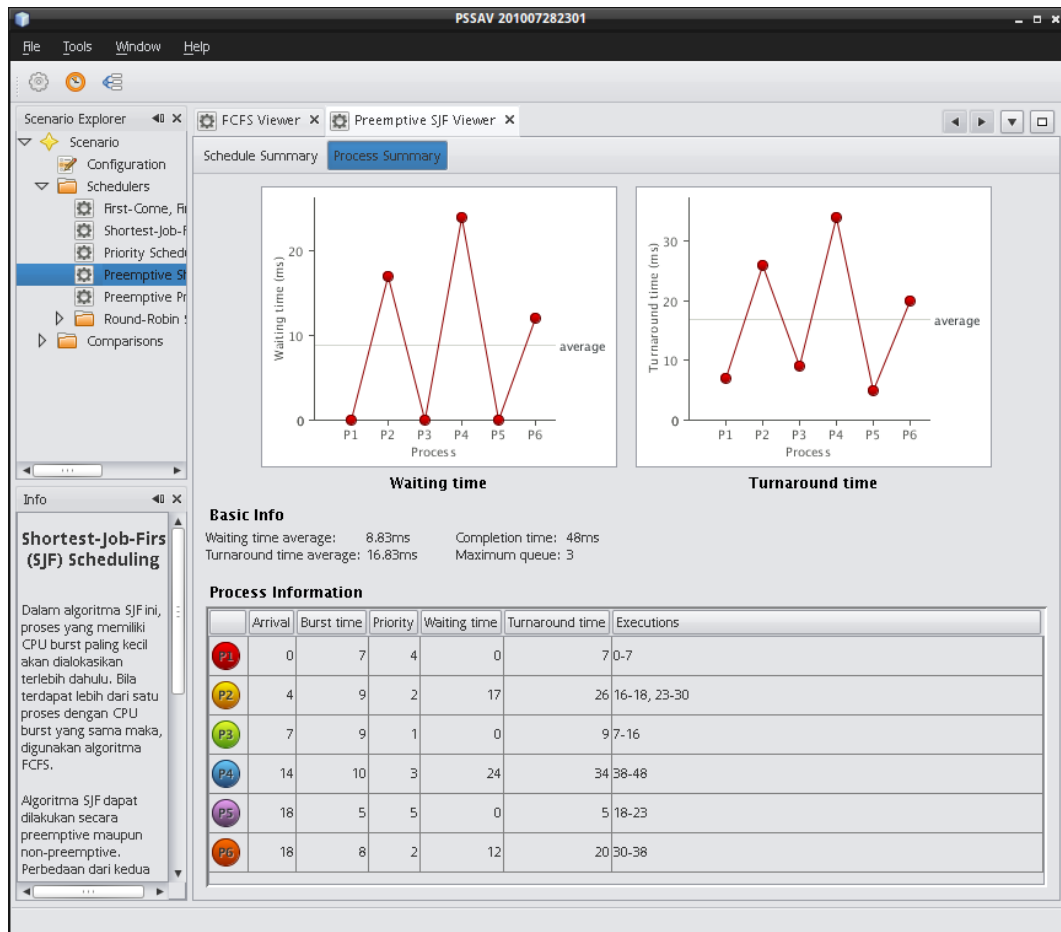
Seperti yang terlihat pada Gambar 6 dan Gambar 7 data simulasi dibagi dalam dua kelompok utama yang ditampilkan dalam *tab*-nya masing-masing yaitu.

**Schedule summary** berisi data umum dari simulasi, data-data pengalokasian CPU yang divisualisasikan ke dalam bentuk animasi *Gantt chart*, serta data kondisi dari proses-proses pada setiap CPU *time* yang ditampilkan dalam bentuk tabel.

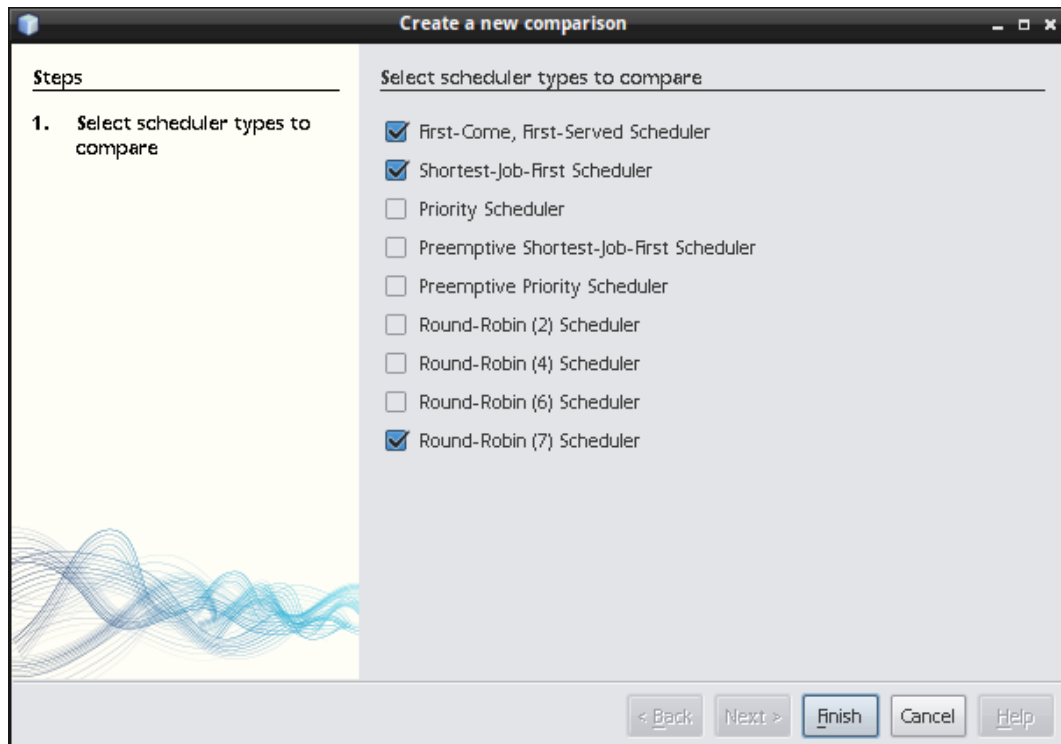
**Process summary** berisi data umum dari simulasi, dan data pengalokasian CPU untuk setiap proses yang ditampilkan dalam bentuk grafik (untuk *waiting time* dan *turnaround time*) dan tabel.



Gambar 6: Tab Schedule Summary pada data simulasi *preemptive SJF*



Gambar 7: Tab Process Summary pada data simulasi *preemptive SJF*



Gambar 8: Wizard New comparison. . .

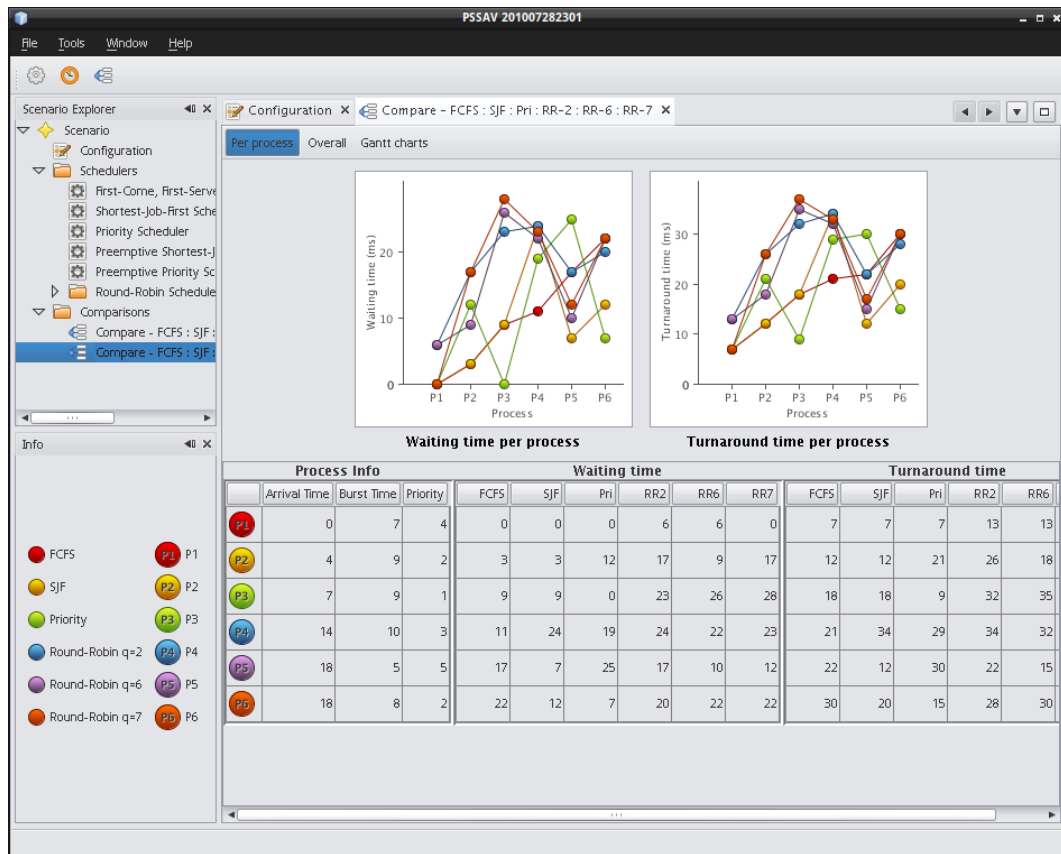
### 3.4.6 Perbandingan Simulasi

Secara *default*, aplikasi PSSAV akan membuatkan satu perbandingan yang membandingkan semua algoritma simulasi yang didukung oleh aplikasi PSSAV ini. Namun kita juga dapat membuat kriteria perbandingan baru yang hanya membandingkan sebagian dari algoritma penjadualan. Untuk melakukan hal tersebut digunakan *wizard New comparison. . .* yang dapat ditampilkan dengan jalan menekan tombol pada *New comparison. . .* toolbar. Tampilan dari *wizard New comparison. . .* dapat dilihat pada Gambar 8. Perbandingan baru akan dapat dibuat apabila telah dipilih lebih dari satu algoritma yang akan dibandingkan.

Sebagai contoh, Gambar 9, 10, dan 11 adalah tampilan perbandingan antara enam buah algoritma penjadualan yaitu FCFS, SJF, *Priority*, dan tiga algoritma *Round-robin* dengan *time quantum* masing-masing adalah 2, 6, dan 7.

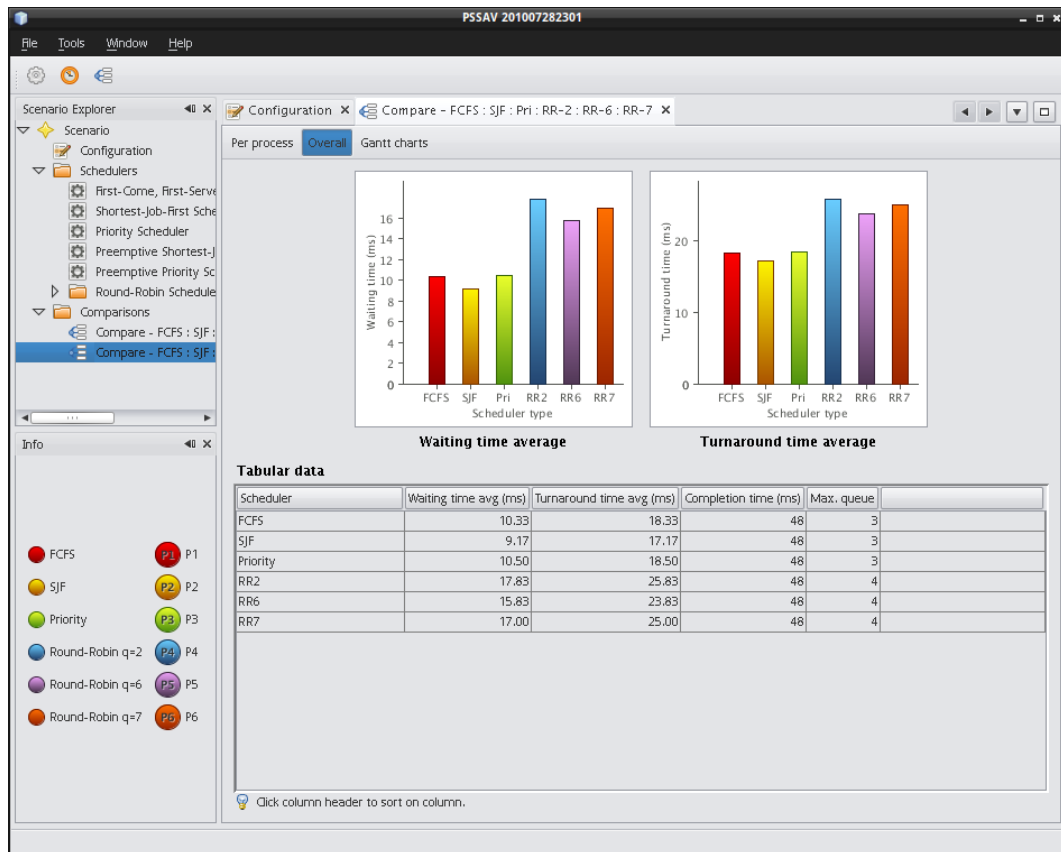
Tampilan perbandingan algoritma dibagi menjadi tiga *tab* yaitu.

- Pada *tab Per Process*, ditampilkan grafik dan tabel perbandingan *waiting time* dan *turnaround time* dari tiap-tiap proses untuk setiap algoritma yang dibandingkan.
- Pada *tab Overall*, ditampilkan grafik dan tabel perbandingan rata-rata *waiting time* dan *turnaround time* untuk setiap algoritma yang dibandingkan.

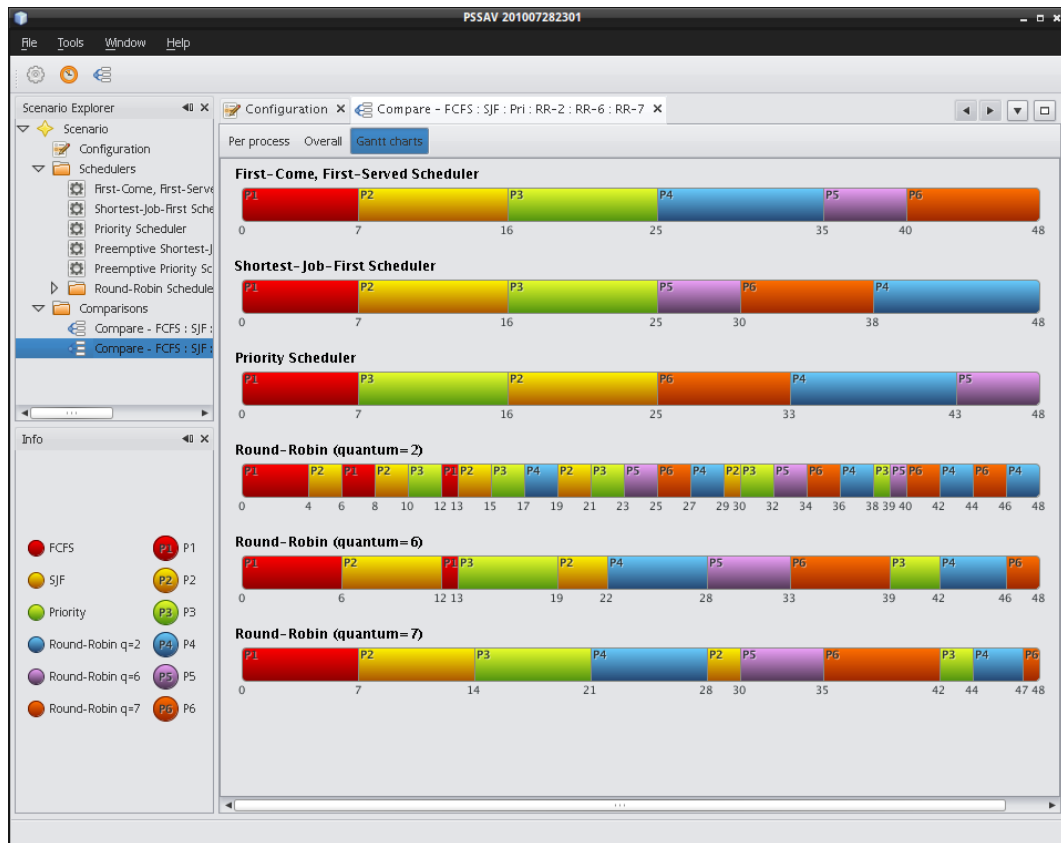


Gambar 9: Tab Per Process pada perbandingan enam buah algoritma penjadualan

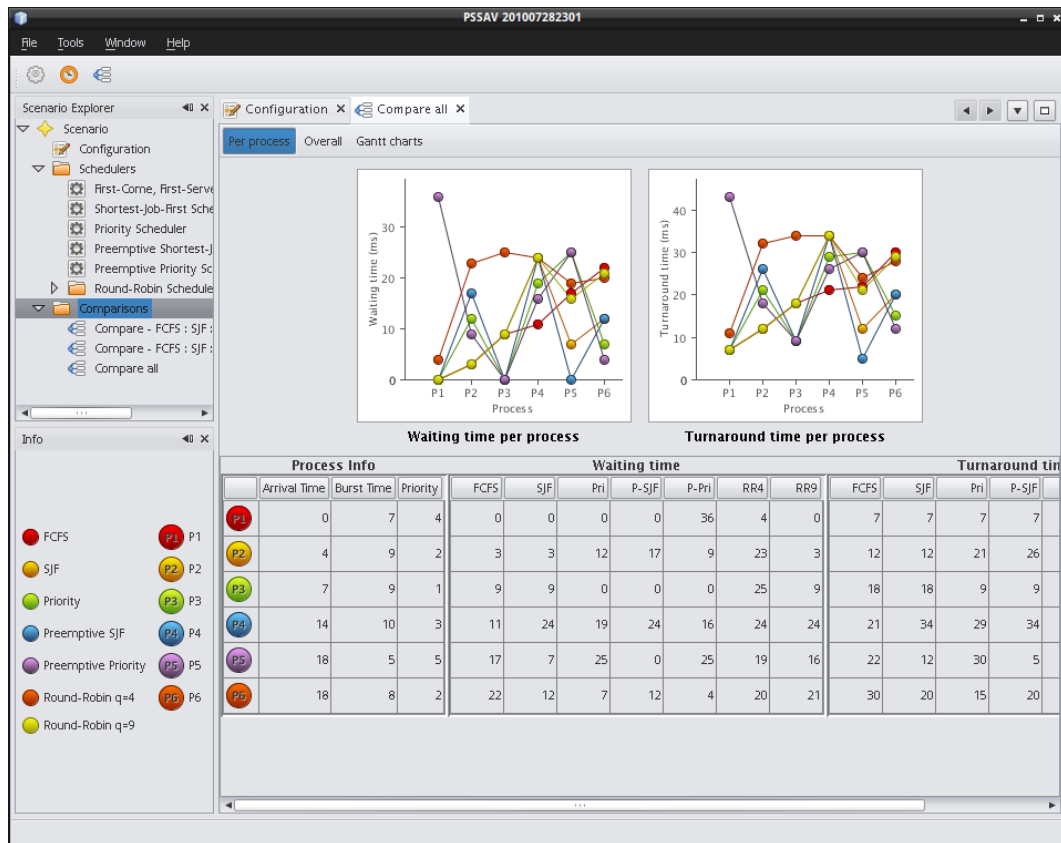
- Pada tab Gantt charts, ditampilkan *Gantt chart* untuk setiap algoritma yang dibandingkan.



Gambar 10: *Tab Overall* pada perbandingan enam buah algoritma penjadualan



Gambar 11: Tab Gantt charts pada perbandingan enam buah algoritma penjadualan



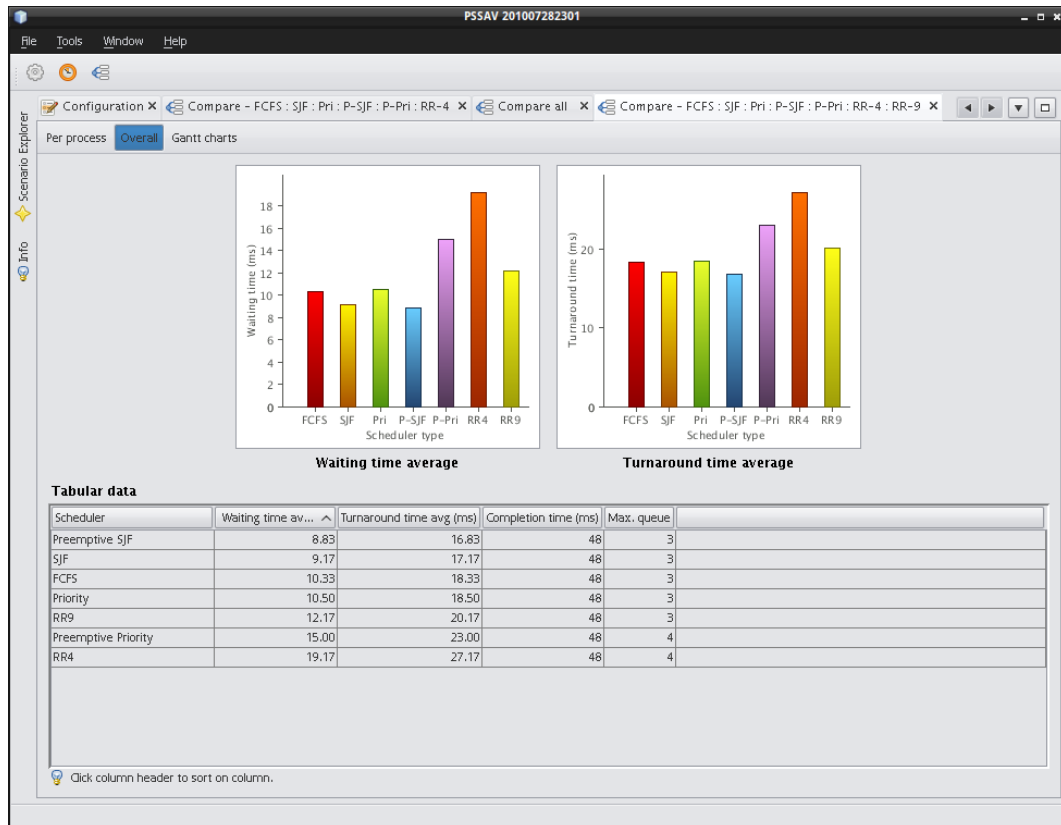
Gambar 12: Tab Per Process pada hasil perbandingan

## 4 Analisis Hasil Simulasi

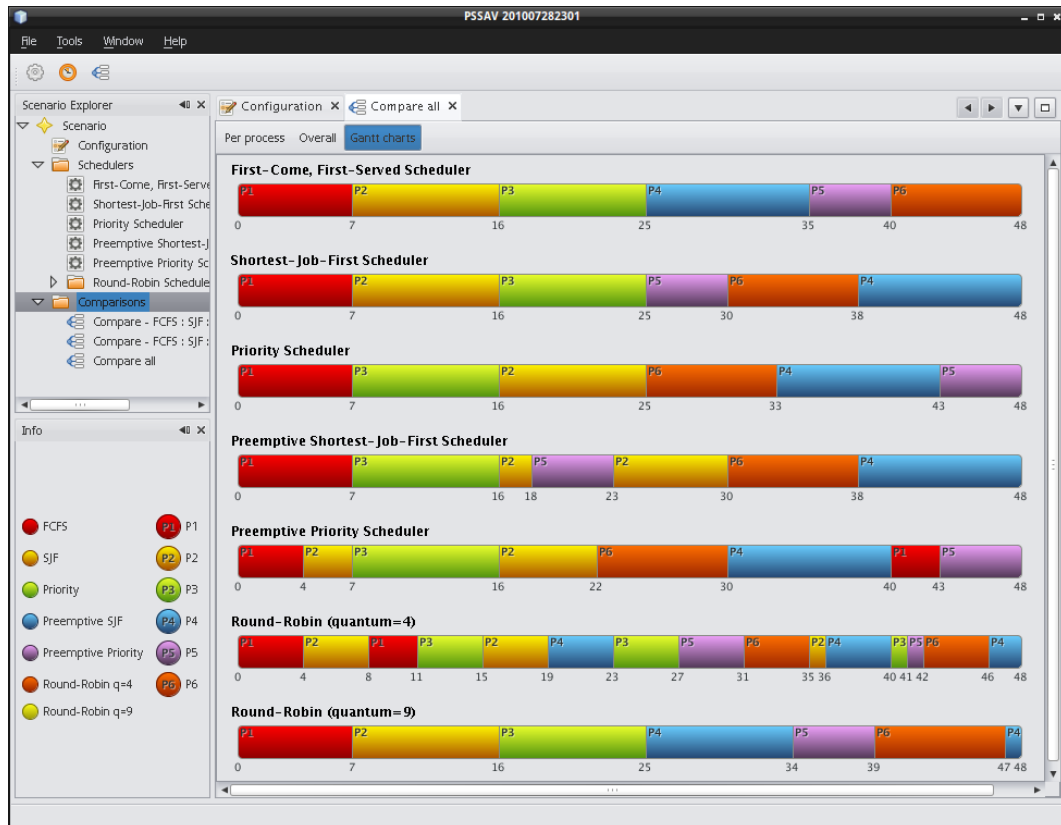
Analisis hasil simulasi dilakukan pada simulasi dengan konfigurasi *default* (sama dengan soal UTS Sistem Operasi tanggal 25 Oktober 2010) hanya saja untuk konfigurasi round-robin digunakan dua buah nilai time quantum yaitu 4 dan 9.

Dari Gambar 12, 13, dan 14 dapat diambil kesimpulan bahwa, dengan konfigurasi proses seperti yang telah dijelaskan, semua algoritma mampu menyelesaikan pengeksekusian proses-prosesnya dalam total waktu yang sama yaitu 48 ms. Rata-rata *waiting time* dan *turnaround time* terkecil dimiliki oleh simulasi algoritma *Preemptive SJF*. Sedangkan rata-rata *waiting time* dan *turnaround time* terbesar dimiliki oleh simulasi algoritma *Round-robin* dengan *time quantum* 4.





Gambar 13: *Tab Overall* pada hasil perbandingan



Gambar 14: *Tab* Gantt chats pada hasil perbandingan

## Pustaka

- [1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 7th ed. John Wiley & Sons, Inc., 2005.