Modul 11 : Penjadwalan Proses dan Manajemen Memori (OS SIM)

Tujuan

a. Mahasiswa mampu memahami proses penjadwalan suatu sistem operasi
b. Mahasiswa mengenal dan menguasai berbagai macam jenis proses penjadwalan
c. Mahasiswa mampu memahami proses manajemen memori
d. Mahasiswa mampu mensetting parameter dalam manajemen memori

Pendahuluan

Pada praktikum ini akan dikenalkan beberapa hal penting yang terkait dengan algoritma penjadwalan proses dan proses manajemen memori. Adapun tool yang akan digunakan yaitu OS Simulator atau disingkat dengan OS Sim.

OS Sim merupakan aplikasi free yang berisi tentang simulasi sistem operasi (SO) secara grafis. Aplikasi ini bertujuan sebagai media pembelajaran yang berfokus pada empat komponen SO diantaranya penjadwalan proses, manajemen memori, manajemen file sistem dan penjadwalan disk

Penjadwalan Proses

Penjadwalan berkaitan dengan permasalahan memutuskan proses mana yang akan dilaksanakan dalam suatu sistem. Proses yang belum mendapat jatah alokasi dari CPU akan mengantri di *ready queue*. Algoritma penjadwalan berfungsi untuk menentukan proses manakah yang ada di *ready queue* yang akan dieksekusi oleh CPU.

Algoritma penjadwalan terdiri dari preemptive dan non-preemptive. Algoritma non-preemptive dibuat agar jika ada proses yang masuk ke running state, maka proses itu tidak bisa diganggu sampai menyelesaikan prosesnya, sedangkan penjadwalan preemptive menggunakan prioritas dimana jadwal dapat menggangu proses dengan prioritas rendah ketika proses dengan prioritas tinggi sedang running.

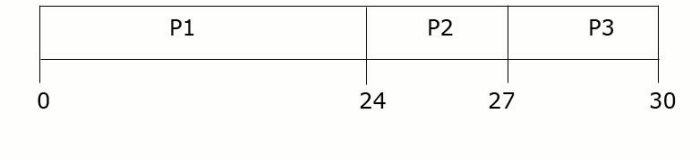
Ada 4 algoritma yang akan dibahas yaitu:

a. FCFS (First Come First Served)

Algoritma ini merupakan algoritma penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma ini setiap proses yang berada pada status *ready* dimasukkan kedalam *FIFO queue* atau antrian dengan prinsip *first in first out*, sesuai dengan waktu kedatangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi.

Contoh:

Ada tiga buah proses yang datang secara bersamaan yaitu pada 0 ms, P1 memiliki burst time 24 ms, P2 memiliki burst time 3 ms, dan P3 memiliki burst time 3 ms. Hitunglah *waiting time* rata-rata dan *turnaround tim e*(*burst time* + *waiting time*) dari ketiga proses tersebut dengan menggunakan algoritma FCFS. *Waiting time* untuk P1 adalah 0 ms (P1 tidak perlu menunggu), sedangkan untuk P2 adalah sebesar 24 ms (menunggu P1 selesai), dan untuk P3 sebesar 27 ms (menunggu P1 dan P2 selesai).



Urutan kedatangan adalah P1, P2 , P3; gantt chart untuk urutan ini adalah:

Waiting time rata-ratanya adalah sebesar(0+24+27)/3 = 17ms. Turnaround time untuk P1 sebesar 24 ms, sedangkan untuk P2 sebesar 27 ms (dihitung dari awal kedatangan P2 hingga selesai dieksekusi), untuk P3 sebesar 30 ms. Turnaround time rata-rata untuk ketiga proses tersebut adalah (24+27+30)/3 = 27 ms.

b. ***Shortest-Job-First(SJF)***

Pada algoritma ini setiap proses yang ada di *ready queue* akan dieksekusi berdasarkan *burst time* terkecil. Hal ini mengakibatkan *waiting time* yang pendek untuk setiap proses dan karena hal tersebut maka *waiting time* rata-ratanya juga menjadi pendek, sehingga dapat dikatakan bahwa algoritma ini adalah algoritma yang optimal.

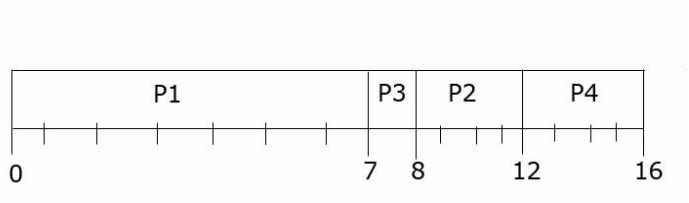
Tabel 14.1. Contoh Shortest Job First

Process	*Arrival Time*	*Burst Time*
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Contoh: Ada 4 buah proses yang datang berurutan yaitu P1 dengan *arrival time* pada 0.0 ms dan *burst time* 7 ms, P2 dengan *arrival time* pada 2.0 ms dan *burst time* 4 ms, P3 dengan *arrival time* pada 4.0 ms dan *burst time* 1 ms, P4 dengan *arrival time* pada 5.0 ms dan *burst time* 4 ms. Hitunglah *waiting time* rata-rata dan *turnaround time* dari keempat proses tersebut dengan mengunakan algoritma SJF.

Average waiting time rata-rata untuk ketiga proses tersebut adalah sebesar (0 +6+3+7)/4=4 ms.

Gambar 14.3. Shortest Job First (Non-Preemptive)



Average waiting time rata-rata untuk ketiga prses tersebut adalah sebesar (9+1+0+2)/4=3 ms

c. ***Priority***

Priority Scheduling merupakan algoritma penjadwalan yang mendahulukan proses yang memiliki prioritas tertinggi. Setiap proses memiliki prioritasnya masing-masing.

Prioritas suatu proses dapat ditentukan melalui beberapa karakteristik antara lain:

1. Time limit.
2. Memory requirement.
3. Akses file.
4. Perbandingan antara burst M/K dengan CPU burst.
5. Tingkat kepentingan proses.

Priority scheduling juga dapat dijalankan secara preemptive maupun non-preemptive. Pada preemptive, jika ada suatu proses yang baru datang memiliki prioritas yang lebih tinggi daripada proses yang sedang dijalankan, maka proses yang sedang berjalan tersebut dihentikan, lalu CPU dialihkan untuk proses yang baru datang tersebut. Sementara itu, pada non-preemptive, proses yang baru datang tidak dapat menganggu proses yang sedang berjalan, tetapi hanya diletakkan di depan queue.

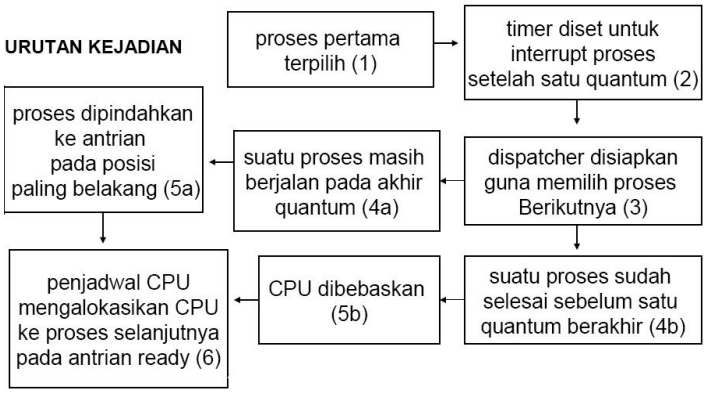
d. ***Round Robin (RR)***

Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar *time quantum*. Jika *time quantum*-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 *quantum*.

Algoritma ini sepenuhnya bergantung besarnya *time quantum*. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma *first come first served*. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang.

Permasalahan utama pada *Round Robin* adalah menentukan besarnya *time quantum*. Jika time *quantum* yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesai dalam 1 *quantum*. Hal ini tidak baik karena akan terjadi banyak *switch*, padahal CPU memerlukan waktu untuk beralih dari suatu proses ke proses lain (disebut dengan context switches time). Sebaliknya, jika time quantum terlalu besar, algoritma *Round Robin* akan berjalan seperti algoritma *first come first served*. *Time quantum* yang ideal adalah jika 80% dari total proses memiliki CPU burst time yang lebih kecil dari 1 *time quantum*.

Gambar 14.4. Urutan Kejadian Algoritma Round Robin



Manajemen Memori

Algoritma manajemen memori dibagi menjadi dua grup, yaitu:

- *Contiguous memory management*, yaitu seluruh proses dialokasikan ke memori. Untuk setiap prosess, partisi yang sedang tidak dipakai dan cukup besar untuk mengakomodasi proses yang membutuhkan akan dipilih.

  Terdapat dua algoritma manajemen memori:

  - Partisi tetap (f*ixed-sized paritions*), memori awalnya dibagi menjadi beberapa partisi dengan ukuran tetap (pengguna yang membuat). Pada umumnya proses yang dialokasikan kedalam sebuah partisi memiliki ukuran yang lebih kecil daripada ukuran partisinya, dan oleh karena itu ada sebagian ukuran partisi yang tidak terpakai, yang disebut sebagai *internal fragmentation*.
  - Partisi tidak tetap (*variable-sized partitions*), memori awalnya merupakan sebuah partisi besar. Lalu untuk setiap proses akan dialokasikan ukuran partisi yang sesuai dengan ukuran proses, dan setelah selesai maka partisi tersebut dapat digunakan ulang. Partisi yang telah selesai digunakan tersebut dapat disebut sebagai *external fragmentation*.

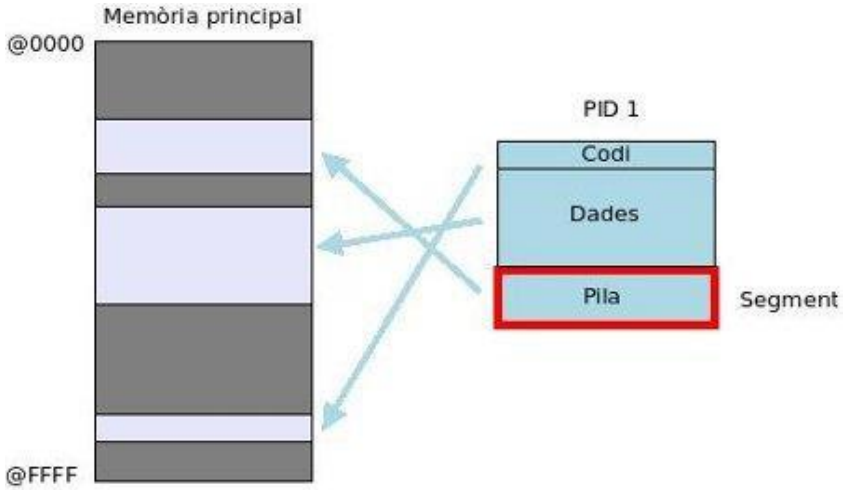
  Konsep lain yang menjadi perhatian adalah aturan alokasi memori, yaitu bagaimana sistem operasi memilih satu dari beberapa partisi yang tersedia. Beberapa diantaranya adalah:

  - *First fit*, memilih partisi pertama kali yang dapat memuat proses.
  - *Best fit,* memilih partisi yang dapat memuat proses secara optimal.
  - Worst fit, memilih partisi yang dapat memuat proses dengan pemilihan paling buruk.

- *Non-contiguous memory management*, yaitu seluruh proses dibagi menjadi bagian-bagian yang dapat dialokasikan secara terpisah di memori, antara lain:

  - *Pagination,* memori dan proses dibagi menjadi bagian-bagian yang berukuran sama (frame dan page), dimana page dari proses akan dialokasikan pada frame dari memori.

- *Segmentation,* proses dibagi menjadi bagian-bagian secara logical (misalnya code, data, atau stack) dan masing-masing akan dialokasikan ke partisi memori secara independen.



Tampilan awal

Main memory

Proses-proses nantinya akan dialokasikan disini, baik itu secara keseluruhan maupun per bagian page atau segmen.

Secara definisi, pada setiap sistem manajemen memori, akan ada selalu terdapat satu proses pertama yang akan dialokasikan ke memori yang mendapatkan alamat dasar (base addresses), yaitu Operating System, yang ukurannya bervariasi dapat diantara 1, 2, atau 4 unit.

Alamat, proses, partisi yang tersedia maupun partisi yang sedang digunakan akan selalu terlihat, dan juga fragmentasi mengikuti kode warna yang memungkinkan pemahaman mahasiswa lebih cepat.

Warna	**Keterangan**
Abu-abu	Operating System
Putih	Memori belum digunakan
Titik biru	Fragmentasi internal
Titik merah muda	Fragmentasi eksternal
Warna lain...	Proses (masing-masing)

Ukuran memori dari simulasi ini bervariasi antara 64 sampai dengan 256 unit.

Klik kanan pada memori untuk menampilkan **pop up menu** yang memungkinkan untuk melakukan beberapa pilihan:

- Saat simulasi berhenti (hanya untuk partisi tetap) >> untuk update dan delete partisi
- Saat waktu simulasi untuk setiap proses yang dialokasikan ke memori >> delete proses, swap proses, melihat address translation, melihat informasi detail (hanya untuk pagination dan segmentation) dan untuk melakukan defragment memori (hanya untuk partisi tidak tetap dan segmentation).

Process queue

Memuat proses yang tersedia untuk dialokasikan ke memori diurutkan berdasarkan waktu kedatangan proses.

Untuk setiap proses menunjukkan informasi yang relevan (PID, nama, dan durasi), disamping dari ukuran dan distribusinya, dimana setiap kotak bersesuaian ke satu unit space, yang dapat digunakan oleh satu alamat memori.

Pada pagination dan segmentation, proses-proses dibedakan menjadi komponen-komponennya, dan di-highlight oleh warna abu-abu untuk komponen yang belum dialokasikan ke memori tetapi dialokasikan ke swap area.

Saat simulasi berhenti, klik kanan pada sebarang proses untuk melihat pop up menu yang memungkinkan untuk meng-update dan delete.

Settings

- *Memory Size* (Ukuran memori, bervariasi antara 64 s/d 256 unit).
- *Operating System Size* (Ukuran memori untuk OS, bervariasi dari 1, 2 atau 4 unit).
- Algorithm:
    - *Fixed-sized partitions* (contiguous memory management) >> required untuk mempartisi seluruh memori.
    - *Variable-sized partitions* (contiguous memory management).
    - *Pagination* (non-contiguous memory management) >> harus menentukan ukuran page sistem (antara 1, 2 atau 4).
    - *Segmentation* (non-contiguous memory management).
- Allocation policy, memilih algoritma pemilihan partisi, yaitu:
    - *First fit*
    - *Best fit*
    - *Worst fit*

Perubahan pada algoritma, ukuran memori, atau ukuran OS membutuhkan untuk me-restart simulasi dan menghapus seluruh proses yang ada.

Menambah proses

Proses hanya dapat ditambahkan selagi simulasi berhenti, untuk setiap proses baru dapat ditambahkan dengan cara berikut:

- *PID* >> proses identifier, bersifat *automatically calculated* dan *incremental*.
- *Name (required)* >> nama proses.
- *Size* >> ukuran proses (nilai: 1 s/d 64 unit).
- *Duration* >> proses dapat mempunyai durasi tak terhingga (diisi -1) atau terhingga (1-100), tapi tidak boleh 0.
- *Color* >> warna yang akan digunakan untuk menandai proses yang dimaksud.

Tambahan untuk pagination:

- *Page table* >> jumlah page yang tergantung pada ukuran proses, untuk setiap page mahasiswa harus menentukan apakah sudah dialokasikan ke memori secara langsung (intially allocated) atau tidak (swapped).

Tambahan untuk segmentation:

- *Segment table* >> setiap proses mempunyai tiga segmen: code, data, dan stack. Untuk setiap segmen, mahasiswa harus menentukan ukuran dan apakah sudah dialokasikan ke memori (initially allocated) atau tidak (swapped). Ukuran dari jumlahan segmen harus sama dengan ukuran dari proses.

Data dan statistik

Menunjukkan informasi alokasi memori pada setiap waktu simulasi. Informasi ini beragam antara satu algoritma dengan algoritma yang lain.

Contiguous memory management, tabel dengan informasi berikut:

- *Direction,* alamat partisi mula-mula.
- *Size,* ukuran dari partisi.

(jika pada partisi terdapat sebuah proses)

- *PID,* proses ID (warna background menunjukkan warna proses).
- *Name,* nama proses.
- *Size,* ukuran proses.
- *Duration,* durasi proses.

Pada pagination, tabel dengan informasi berikut:

- *Address,* alamat frame mula-mula.
- *Frame,* nomor frame.

(jika partisi sedang digunakan)

- *PID,* proses ID (warna background menunjukkan warna proses).
- *Page,* proses yang menggunakan frame yang bersesuaian.
- *Name,* nama proses.
- *Dimensions,* total proses.
- *Duration,* durasi proses.

Pada segmentation, tabel dengan informasi berikut:

- *Address,* alamat frame mula-mula.
- *Size,* ukuran partisi.

(jika pada partisi terdapat sebuah proses)

- *PID,* proses ID (warna background menunjukkan warna proses).
- *Segment,* segmen proses, yaitu: code, data atau stack.
- *Name,* nama proses.
- *Size,* ukuran proses.
- *Duration,* durasi proses.

Tabel page dan tabel segment

Menunjukkan informasi alokasi memori dari sebuah proses pada non-contiguous memory management.

Pada pagination, konten tabel adalah sebagai berikut.

- *Page,* proses yang menggunakan frame yang bersesuaian.

- *Frame,* nomor frame.
- *Valid,* bit valid dari page (v valid, i invalid). Sebuah page dikatakan tidak valid ketika page tersebut tidak dialokasikan ke memori.

Pada segmentation, konten tabel adalah sebagai berikut.

- *Segment,* segmen dari proses: code, data, atau stack.
- *Size,* ukuran dari segmen.
- *Address,* alamat frame mula-mula.
- *Valid,* bit valid dari segmen (v valid, i invalid). Sebuah segmen dikatakan tidak valid ketika segmen tersebut tidak dialokasikan ke memori.

Membuat partisi (untuk partisi berukuran tetap)

Menentukan alamat mula-mula dan ukuran partisi.

Partisi tidak bisa saling overlap, dan tidak bisa melebihi batas akhir memori.

Defragment memori (untuk partisi berukuran tidak tetap dan segmentation)

Paritisi berukuran tidak tetap dan segmentation mempunyai kesamaan pada partisi yang dibuat secara dinamis dan diukur secara otomatis untuk mengakomodasi proses atau segmen secara tepat. Operating System memilih sebuah partisi yang sama atau sedikit lebih besar dan dibagi menjadi dua bagian, satu untuk mengalokasikan proses atau segmen, satu untuk yang masih tersedia.

Setelah selesai digunakan (proses sudah selesai atau swapped out), partisi tersebut menjadi kosong dan berubah menjadi lebih kecil (memorinya ter-degradasi).

Proses defragmentasi dapat menanggulangi untuk me-reorganize memori, mengatur semua proses menjadi menggunakan memori dengan alamat rendah (lower memory addresses) dan menggabungkan seluruh partisi kosong menjadi satu partisi kosong yang besar.

Ini adalah tanggung jawab dari OS, dan ini diserahkan kepada mahasiswa untuk simulasi (pop up menu dari memori).

Swap

Melakukan swap dapat memungkinkan kita untuk menjalankan proses yang lebih banyak dari kapasitas memori utama, dan biasanya diimplementasikan pada memori sekunder.

Hal ini muncul atas dasar untuk mempunyai space ekstra untuk memindah proses atau bagian darinya (page atau segmen) yang sedang kurang aktif dan mengalokasikan ulang ke memori saat sedang dibutuhkan.

Simulasi digunakan untuk menyederhanakan proses ini, mensimulasikan swap in dan swap out antar memori dan swap dilakukan oleh mahasiswa secara manual.

Swap out, dari pop up menu memori (klik kanan pada memori). Melakukan swap out sebuah proses atau bagian darinya.

Swap in, dari pop up menu memori (klik kanan pada memori). Mencoba untuk melakukan swap in pada sebuah proses atau bagian darinya kepada memori, jika memori penuh akan menghasilkan sebuah error dan tidak akan bisa swap in.

Algoritma pagination dan segmentation memungkinkan untuk melakukan swap out page atau segmen pada awal (initial). Area ini tidak dibatasi pada jumlah atau ukuran dari proses yang dimuat.

Error pada simulasi

1. **Memory full,** karena sedang memproses antrian proses pada simulasi, memori mengalokasikan untuk proses dan selesai mengalokasikannya. Saat proses tidak bisa dialokasikan karena kurang space-nya, maka menyebabkan tidak dapat dilanjutkan ke proses

selanjutnya. Memori dapat dilepaskan (released) secara otomatis (karena proses sudah selesai menggunakannya) atau oleh campur tangan user (delete atau swap out proses).

2. **Memory not fully partitioned,** algoritma paritisi berukuran tetap (fixed-size partition) membutuhkan semua memori untuk dipartisi (alamat akhir satu partisi adalah alamat awal partisi yang lain).

Peralatan

- Software: OSSIM
- Hardware: PC desktop/laptop
- Modul Praktikum

Langkah Kerja

Kegiatan 1. Penjadwalan Proses

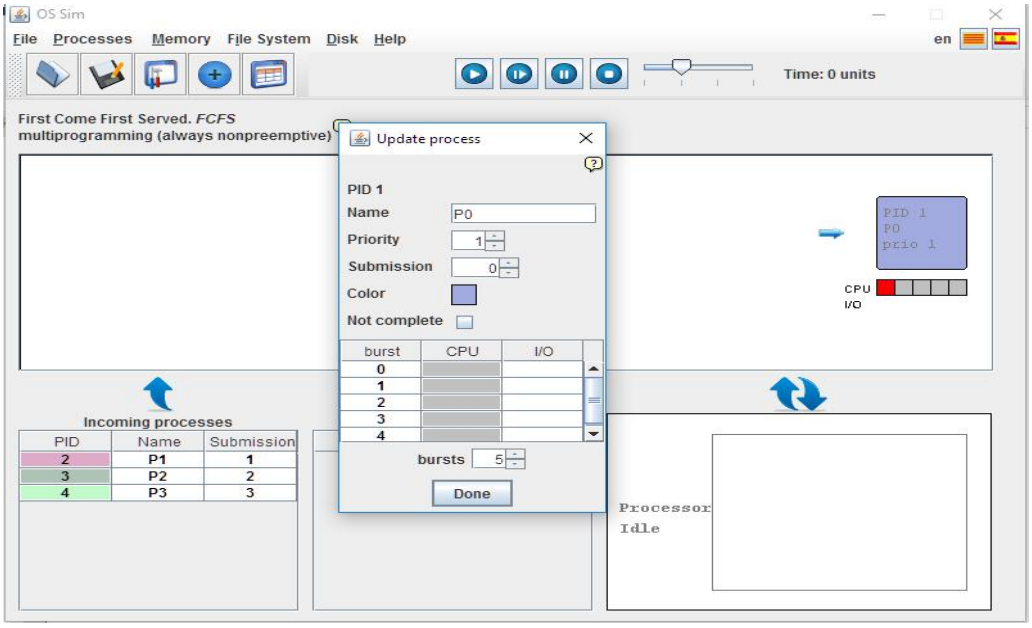
1.1 First-Come, First-Served (FCFS)

a. Bukalah program OSSim, selanjutnya pilih menu processes -> process scheduling

b. Selanjutnya pilihlah setting dan pilih algoritma **First-Come, First-Served (**FCFS)

c. Lakukan input proses sesuai dengan tabel berikut dengan memulai dengan P0 sebagai input proses yang pertama.

Process	Arrival Time	Burst Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



d. Jika input sudah selesai dilakukan. Pilih tombol start pada bagian atas. Amati dan analisa proses yang terjadi.

e. Isilah tabel berikut

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

1.2 **Shortest Job First (SJF)**

a. Bukalah program OS Sim, selanjutnya pilih menu processes -> process scheduling

b. Selanjutnya pilihlah setting dan pilih algoritma **Shortest Job First (SJF).** algoritma ini terdiri dari 2 jenis yaitu non-preemtive dan preemptive. Untuk mengaktifkan preemptive dengan mencentang menu tersebut. Sebaliknya jika menonaktifkan maka hanya cukup menghilangkan centangnya saja.

c. Selanjutnya klik tombol start. Amati dan analisa proses yang terjadi. Lakukan perbandingan dari hasil keduanya.

d. Isilah tabel berikut:

Non-Preemtive

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

Preemtive

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

1.3 **Priority**

a. Pilihlah menu setting dan pilih algoritma **Priority.** Selanjutnya tambahkan priority pada setiap proses.

Process	Arrival Time	Burst Time	Priority	Service Time

P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

b. Selanjutnya klik tombol start. Lakukan pengamatan dan analisa proses yang terjadi.Lengkapilah tabel berikut!

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

1.4 **Round Robin**

a. Pilihlah menu setting dan pilih algoritma **Round Robin.** Selanjutnya tambahkan quantum time sebesar 3.

b. Selanjutnya klik tombol start. Lakukan pengamatan dan analisa proses yang terjadi. Lengkapilah tabel berikut:

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

Kegiatan 2: Manajemen Memori

2.1 Contiguous memory management dengan menggunakan partisi berukuran tetap (fixed-size partition) dan aturan first fit

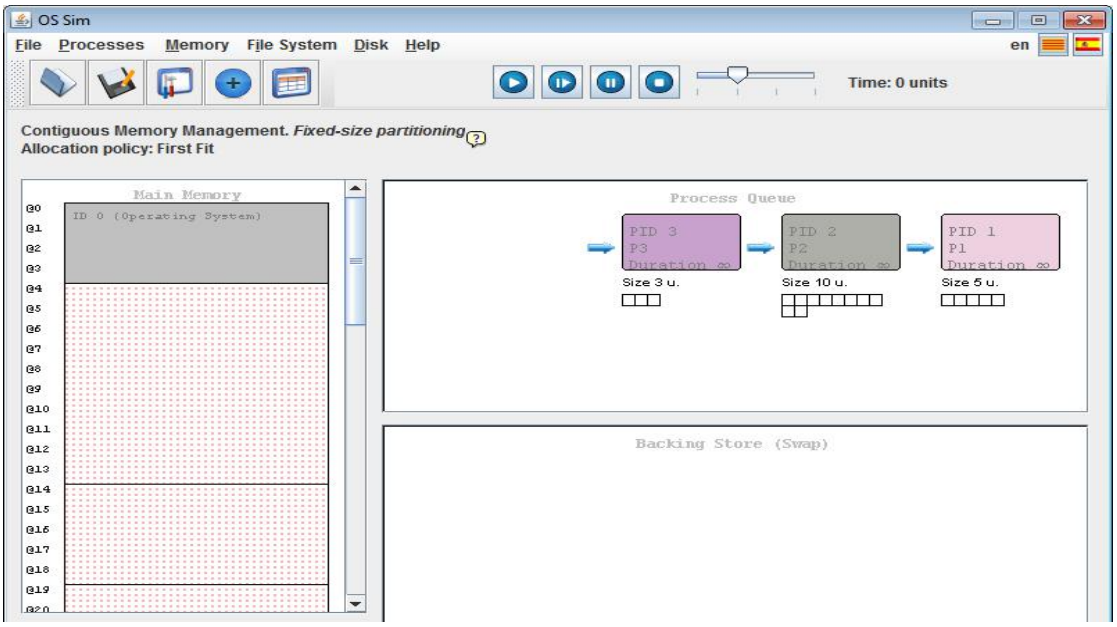
Memori dibagi menjadi partisi-partisi berukuran tetap, lalu seluruh proses akan dialokasikan pada satu dari partisi yang tersedia. Pada aturan first fit, partisi pertama yang dapat memuat proses akan dipilih.

Partisi		Proses	
Alamat awal	Unit ukuran	Proses	Unit ukuran
@4	10	P1	5
@14	5	P2	10
@19	4	P3	3
@23	15		
@38	10		
@48	14		

10

Langkah:

1. Pilih menu help >> examples...>> memory management >> Contiguous memory management with fixed-size partitions (first adjustment). Pilihlah folder hingga menampilkan tampilan berikut.



2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.2 Contiguous memory management dengan menggunakan partisi berukuran tetap (fixed-size partition) dan aturan best fit

Memori dibagi menjadi partisi-partisi berukuran tetap, lalu seluruh proses akan dialokasikan pada satu dari partisi yang tersedia. Pada aturan best fit, partisi terbaik (yang ukurannya sama atau hampir sama) yang dapat memuat proses akan dipilih.

Partisi		Proses	
Alamat awal	Unit ukuran	Proses	Unit ukuran
@4	10	P1	5
@14	5	P2	10
@19	4	P3	3
@23	15		
@38	10		
@48	14		

Langkah:

1. Pilih menu help >> examples...>> memory management >> Contiguous memory management with fixed-size partitions (best fit)>> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.3 Contiguous memory management dengan menggunakan partisi berukuran tidak tetap (variable-size partition) >> defragmentasi

Memori pertama tidak dipartisi, lalu partisi akan dibuat saat proses dialokasikan dan mempunyai ukuran yang sama dengan ukuran proses. Selanjutnya partisi akan dibebaskan (saat proses sudah selesai menggunakan memori) dan digunakan oleh proses lain. Jika ukuran proses lebih kecil dari ukuran partisi, maka partisi akan dibagi menjadi dua bagian, bagian pertama sama besarnya dengan proses dan bagian kedua adalah sisanya.

Proses	Unit ukuran	Durasi
P1	5	5
P2	8	4
P3	3	3
P4	15	∞
P5	2	1
P6	20	∞
P7	3	∞
P8	10	∞

Langkah:

1. Pilih menu help >> examples...>> memory management >>Contiguous memory management with variable-sized partitions (Defragmentation) >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.4 Contiguous memory management dengan menggunakan partisi berukuran tidak tetap (variable-size partition) >> swap

Memori pertama tidak dipartisi, lalu partisi akan dibuat saat proses dialokasikan dan mempunyai ukuran yang sama dengan ukuran proses. Selanjutnya partisi akan dibebaskan (saat proses sudah selesai menggunakan memori) dan digunakan oleh proses lain. Jika ukuran proses lebih kecil dari ukuran partisi, maka partisi akan dibagi menjadi dua bagian, bagian pertama sama besarnya dengan proses dan bagian kedua adalah sisanya.

Memori mempunyai ukuran yang berhingga, hanya beberapa proses saja yang bisa dimuat di memori pada saat yang sama, tetapi space swap memungkinkan kita untuk menyimpan beberapa proses yang sedang kurang aktif, sehingga space pada memori dapat dialokasikan untuk proses yang lain yang lebih aktif. Proses yang telah di swap out akan dialokasikan kembali ke memori saat dibutuhkan.

Proses	Unit ukuran
P1	5
P2	8
P3	3
P4	15
P5	2

P6	20
P7	3
P8	10

Langkah:

1. Pilih menu help >> examples...>> memory management >>Contiguous memory management with variable-sized partitions (Swap) >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.5 Pagination (ukuran page 2 unit)

Proses akan dibagi menjadi beberapa page dengan ukuran tetap (contohnya adalah 2 unit), memori lalu dibagi menjadi beberapa frame berukuran sama. Page dari proses lalu dialokasikan pada frame memori yang kosong.

Proses	Ukuran	Durasi
P1	5 unit (3 page)	4
P2	4 unit (2 page)	∞
P3	11 unit (6 page)	2
P4	5 unit (3 page)	∞
P5	7 unit (4 page)	∞
P6	3 unit (2 page)	∞

Langkah:

1. Pilih menu help >> examples...>> memory management >>Pagination) >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.6 Segmentation (alokasi parsial)

Proses dibagi menjadi beberapa segmen, tiap segmen bisa mempunyai ukuran yang berbeda. Segmen ini akan dialokasikan ke memori secara independen, lalu partisi akan dibuat untuk menampung segmen tersebut dengan ukuran yang sama. Pembagian ini adalah secara fungsional dan berdasarkan pada logical structure dari proses tersebut (data, stack, code, dll).

Tidak semua segmen harus dimuat pada memori agar proses dapat berjalan, selama sebuah segmen tidak dipakai maka dapat di-swap out.

Proses	Ukuran total	Segmen	Ukuran (unit)	Dialokasikan diawal
P1	20	Code	2	Yes
		Data	10	Yes
		Stack	8	No
P2	20	Code	5	Yes
		Data	14	No
		Stack	1	Yes
P3	40	Code	10	Yes
		Data	20	Yes
		Stack	10	Yes

Langkah:

1. Pilih menu help >> examples...>> memory management >>Segmentasi >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

Tugas:

1. Jawab pertanyaan pada latihan OS-SIM scheduling algorithm performance comparison

2. Jawab pertanyaan pada latihan OS-SIM management memory

Modul 12 : Manajemen File System dan Penjadwalan Disk