**PRAKTIKUM SISTEM OPERASI MODUL 11**

**Disusun Oleh :**

**RAMA AZIZ WIBOWO**

**L200210263KELAS E**

**TEKNIK INFORMATIKA**

**FAKULTAS KOMUNIKASI DAN INFORMATIKA UNIVERSITAS MUHAMMADIYAH SURAKARTA 2021/2022**

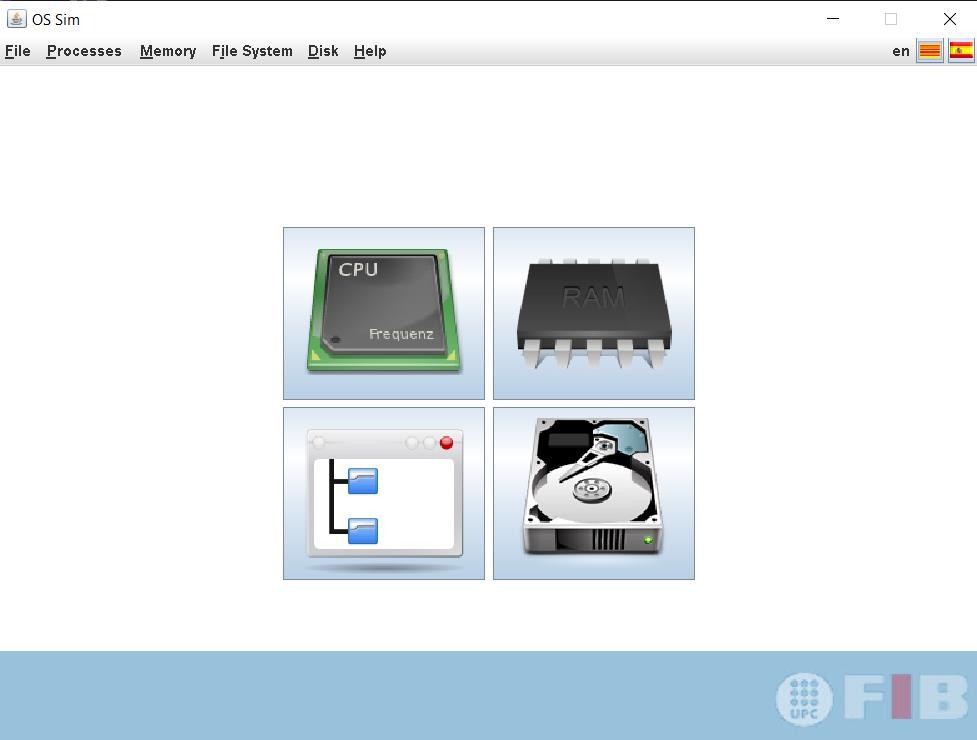
# Kegiatan 1

**First-Come, First-Served (FCFS)**

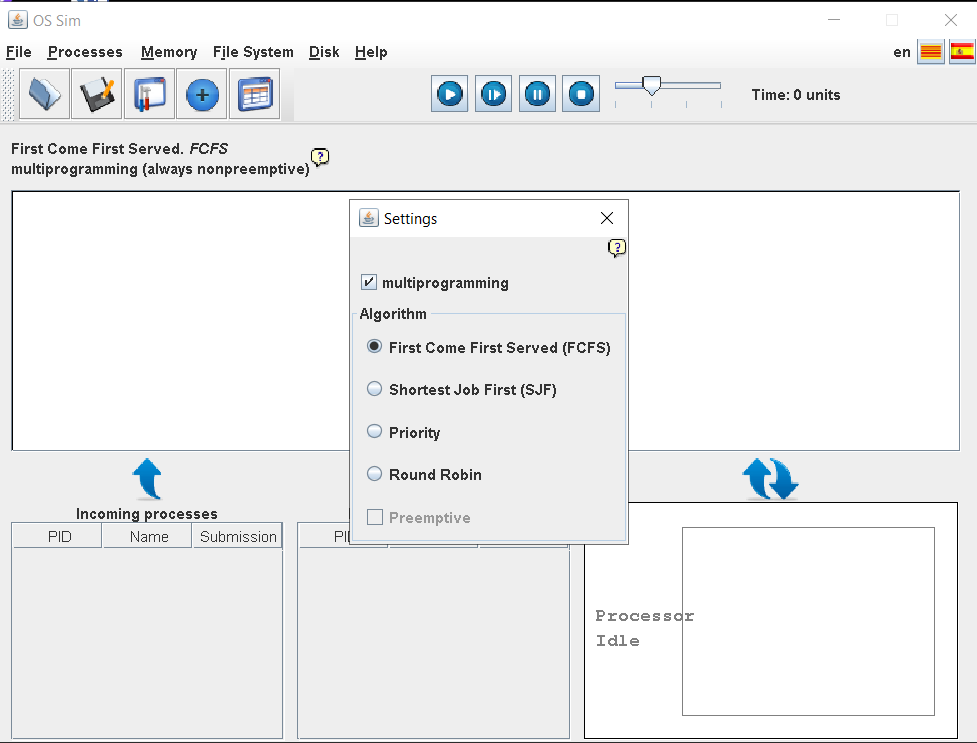
1. Membuka program OSSim, selanjutnya memilih menu processes -> process scheduling
2. Selanjutnya memilih setting dan pilih algoritma First-Come, First-Served (FCFS)
3. Melakukan input proses sesuai dengan tabel berikut dengan memulai dengan P0 sebagai input proses yang pertama.

|  |  |  |  |
| --- | --- | --- | --- |
| Proses | Arrival time | Burst time | Service time |
| **P0** | 0 | 5 | 0 |
| **P1** | 1 | 3 | 5 |
| **P2** | 2 | 8 | 8 |
| **P3** | 3 | 6 | 16 |

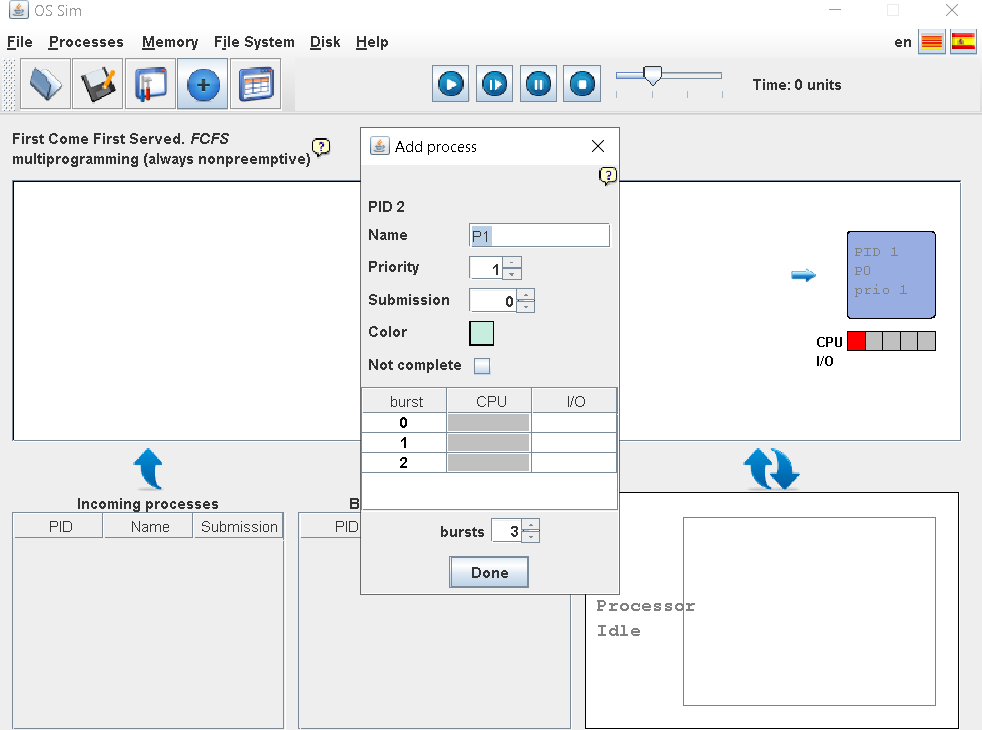
* + Membuka program OSSim.jar

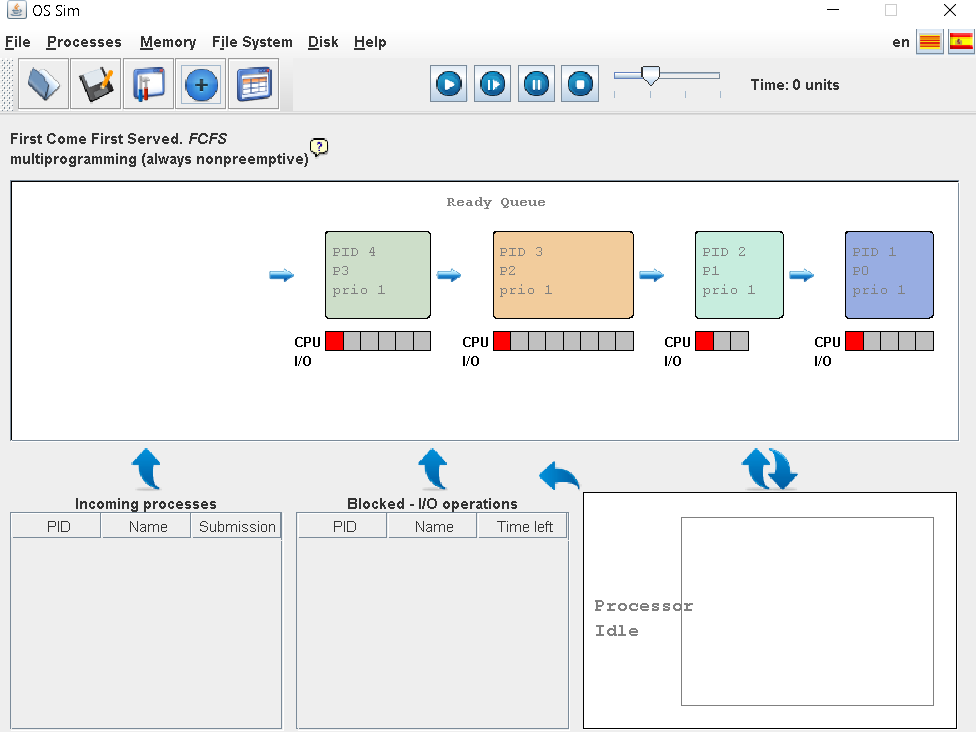


* + Memilih algoritma FCFS

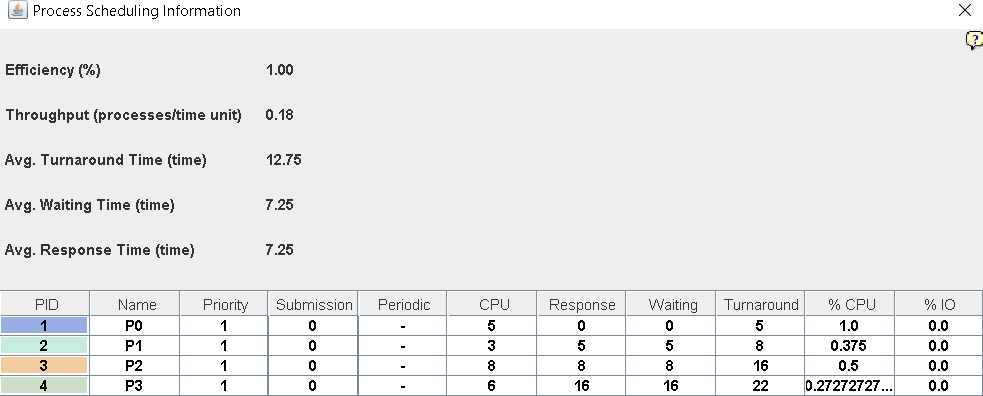


* + Proses Penginputan





* + Output

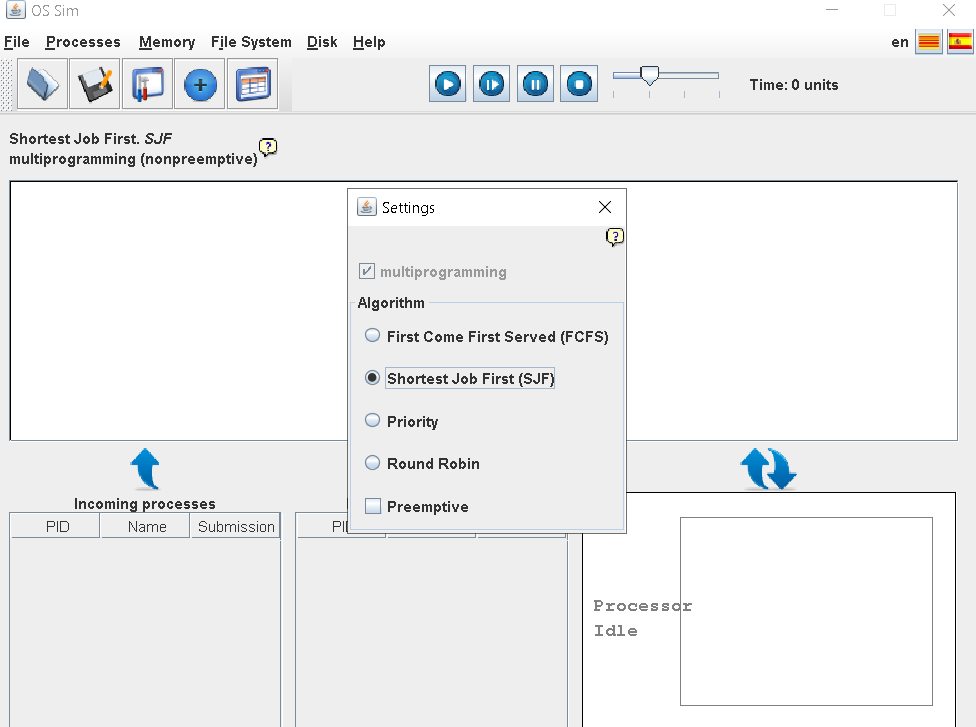


* + Tabel :

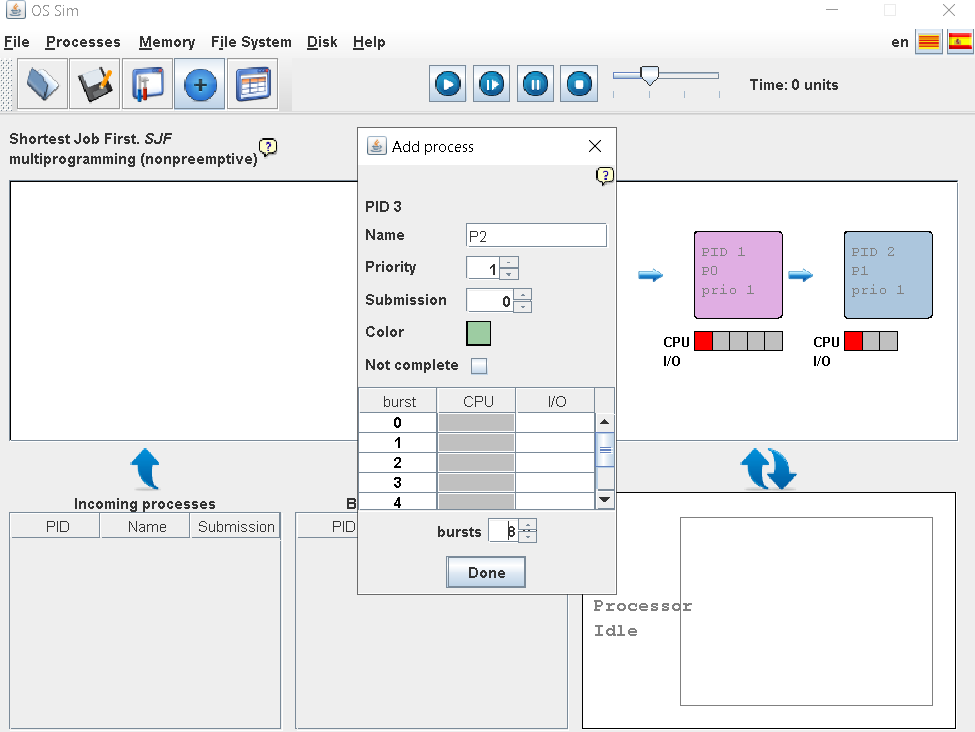
|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time-Arrival Time** |
| P0 | 0 |
| P1 | 5 |
| P2 | 8 |
| P3 | 16 |
| Av Wait Time | 7.25 |

# Shortest Job First (SJF)

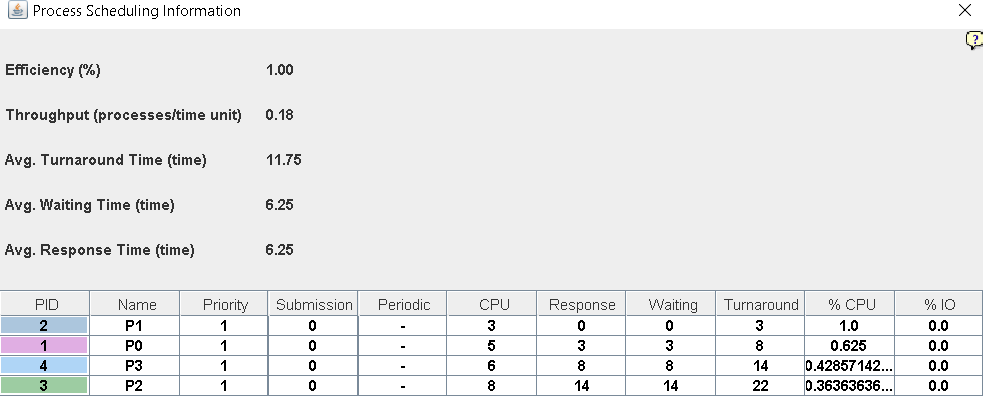
1. Membuka program OS Sim, selanjutnya memilih menu processes -> process scheduling
2. Selanjutnya memilih setting dan pilih algoritma Shortest Job first (SJF). Algoritma ini terdiri dari 2 jenis yaitu non-preemtive dan preemptive. Untuk mengaktifkan preemptive dengan mencentang menu tersebut. Sebaliknya jika menonaktifkan maka hanya cukup menghilangkan centangnya saja.
3. Selanjutnya klik tombol start. Mengamati dan menganalisa proses yang terjadi dan melakukan perbandingan dari hasil keduanya.
   * SJF ( non Preemptive )



* + Proses Penginputan



* + Output

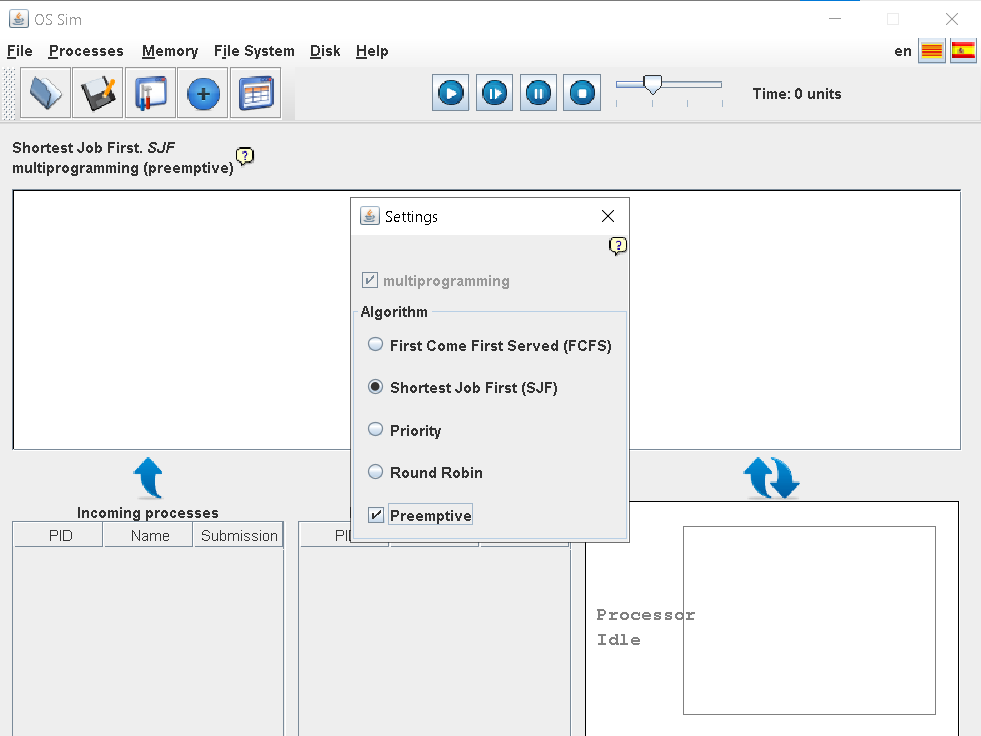


* + Tabel ( nonpreemtive )

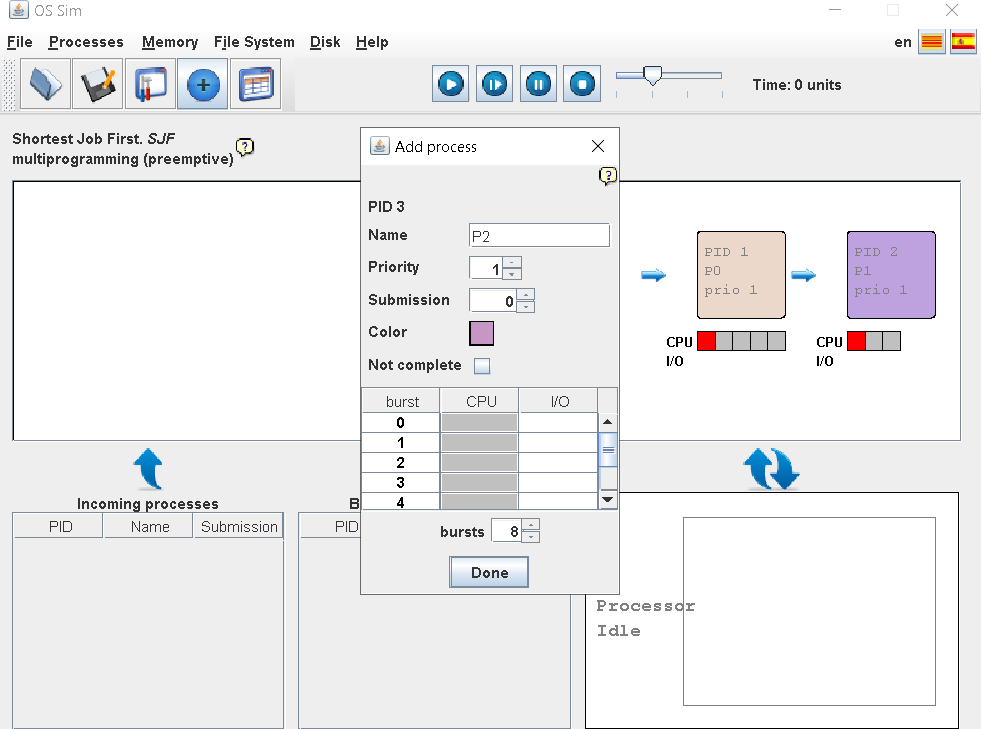
|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time-Arrival Time** |
| P0 | 0 |
| P1 | 3 |
| P2 | 8 |
| P3 | 14 |
| Av Wait Time | 6.25 |

Pada algoritma ini, proses dengan burst time terkecil akan dieksekusi terlebih dahulu, sehingga akan dieksekusi berurutan sesuai dengan besar burst time yang dibutuhkan.

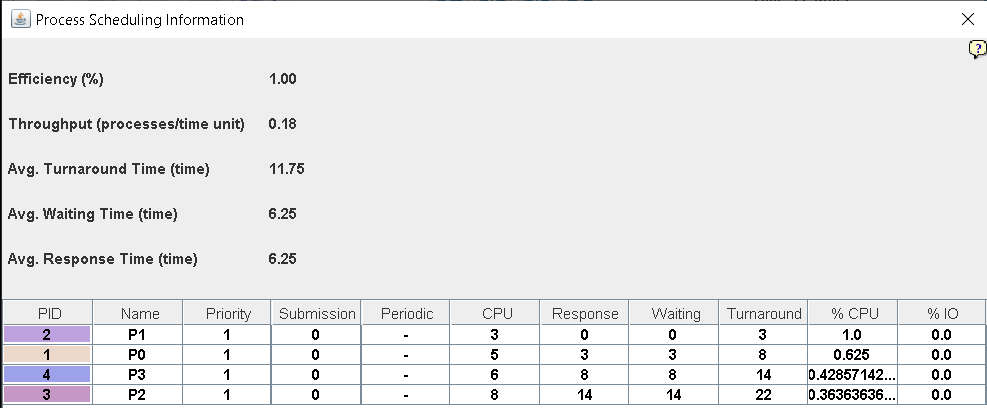
* + SJF ( Preemtive )



* + Proses Penginpputan



* + Output



* + Tabel ( Preemtive )

|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time-Arrival Time** |
| P0 | 0 |
| P1 | 3 |
| P2 | 8 |
| P3 | 14 |
| Av Wait Time | 6.25 |

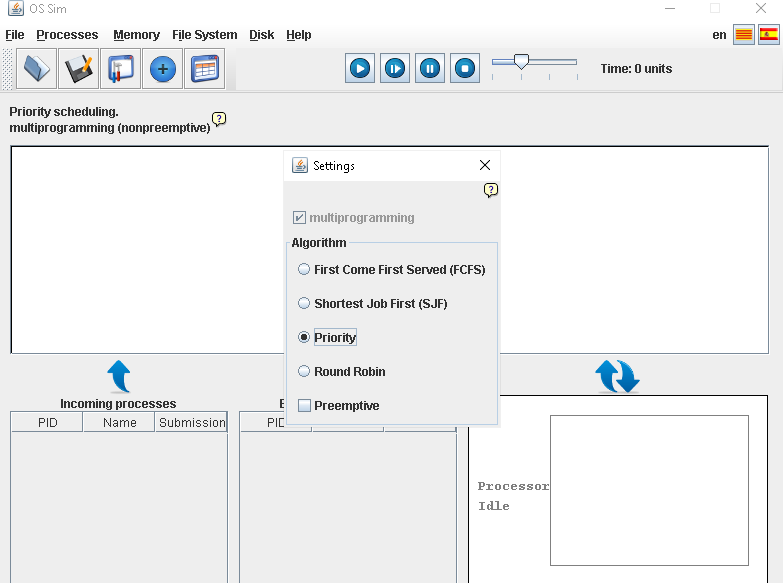
Pada algoritma ini, proses yang memiliki burst time paling sedikit akan dieksekusi terlebih dahulu, dan berlaku pada proses yang pertama kali datang (preemtive) sehingga ratarata waktu tunggu akan kecil.

# Priority

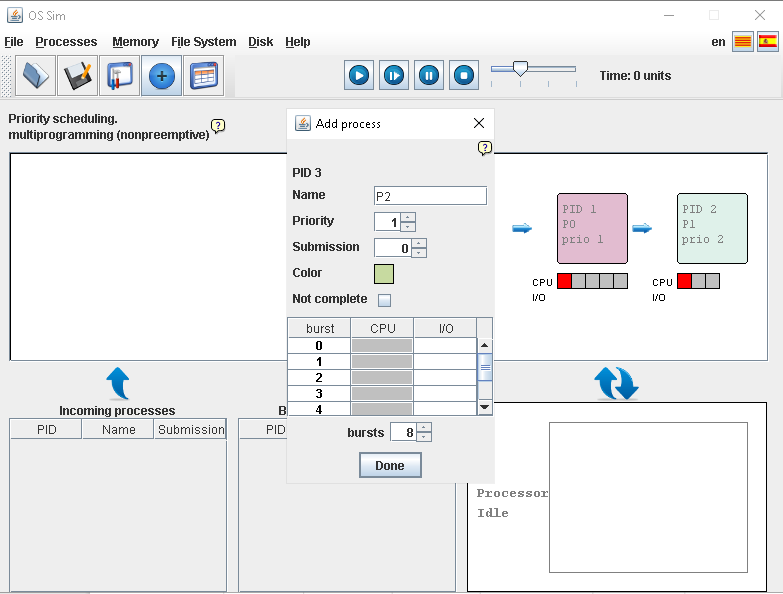
1. Memilih menu setting dan pilih algoritma Priority. Selanjutnya menambahkan priority pada setiap proses.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proses** | **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 |

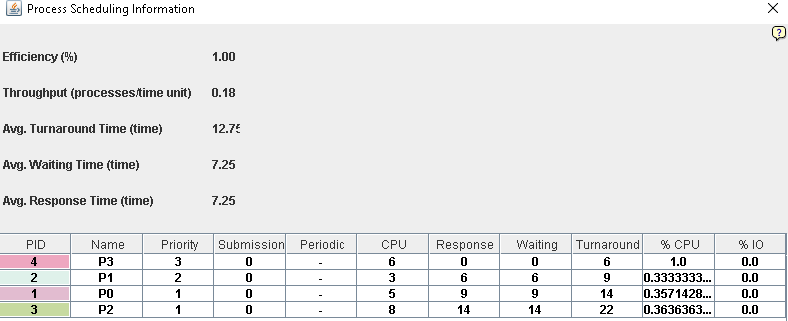
1. Selanjutnya melakukan pengamatan dan Analisa proses yang terjadi.
   * Memilih algoritma Priority



* + Penginputan Proses



* + Output

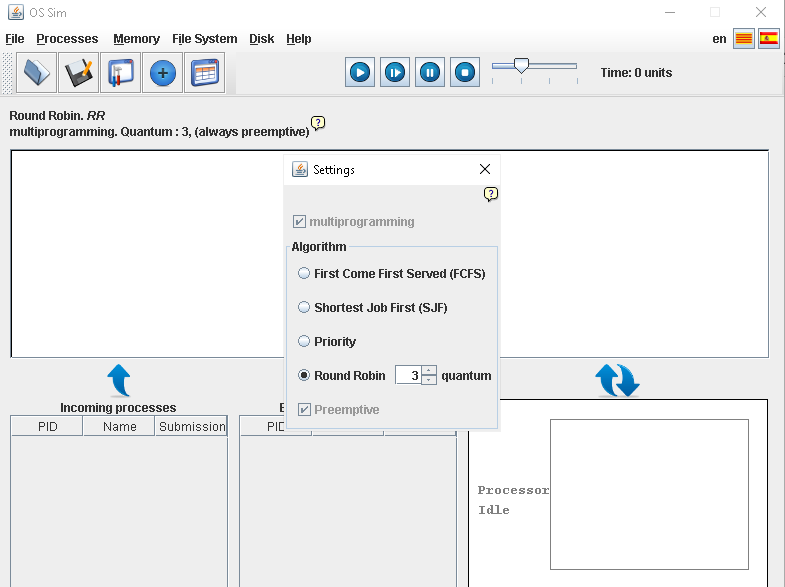


* + Tabel

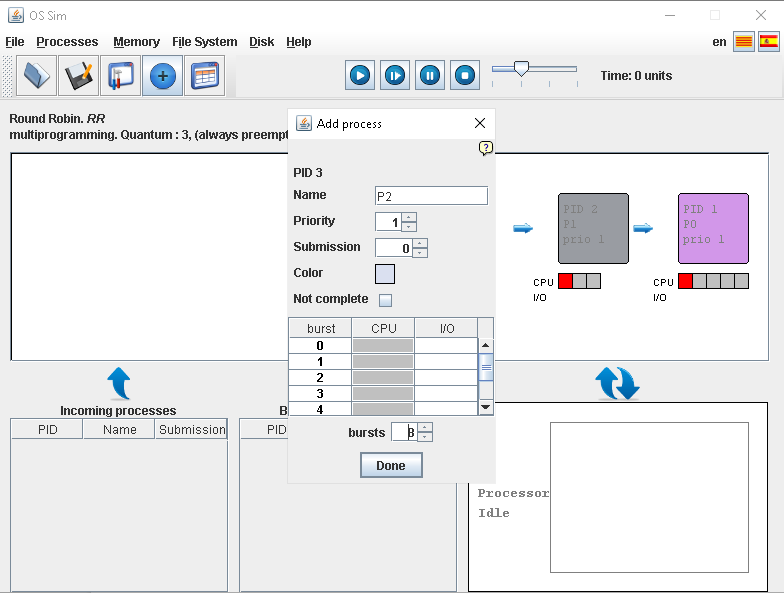
|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time-Arrival Time** |
| P0 | 0 |
| P1 | 6 |
| P2 | 9 |
| P3 | 14 |
| Av Wait Time | 7.25 |

# Round Robin

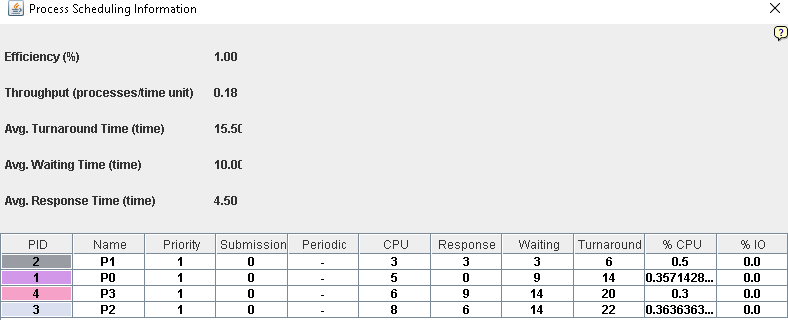
1. Memilih menu setting dan pilih algoritma Round Robin. Selanjutnya menambahkan quantum time sebesar 3.
2. Selanjutnya melakukan pengamatan dan menganalisa proses yang terjadi.
   * Memilih algoritma Round Robin dengan quantum 3



* + Penginputan Proses



* + Output



* + Tabel

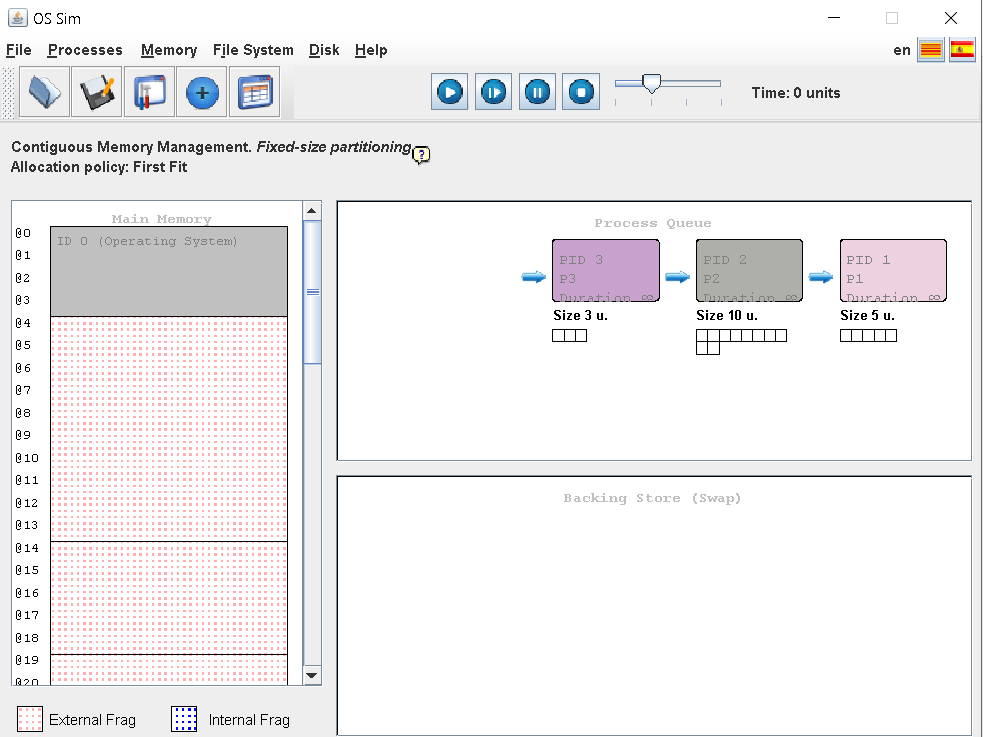
|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time-Arrival Time** |
| P0 | 3 |
| P1 | 9 |
| P2 | 14 |
| P3 | 14 |
| Av Wait Time | 10.00 |

# Kegiatan 2. Manajemen Memori

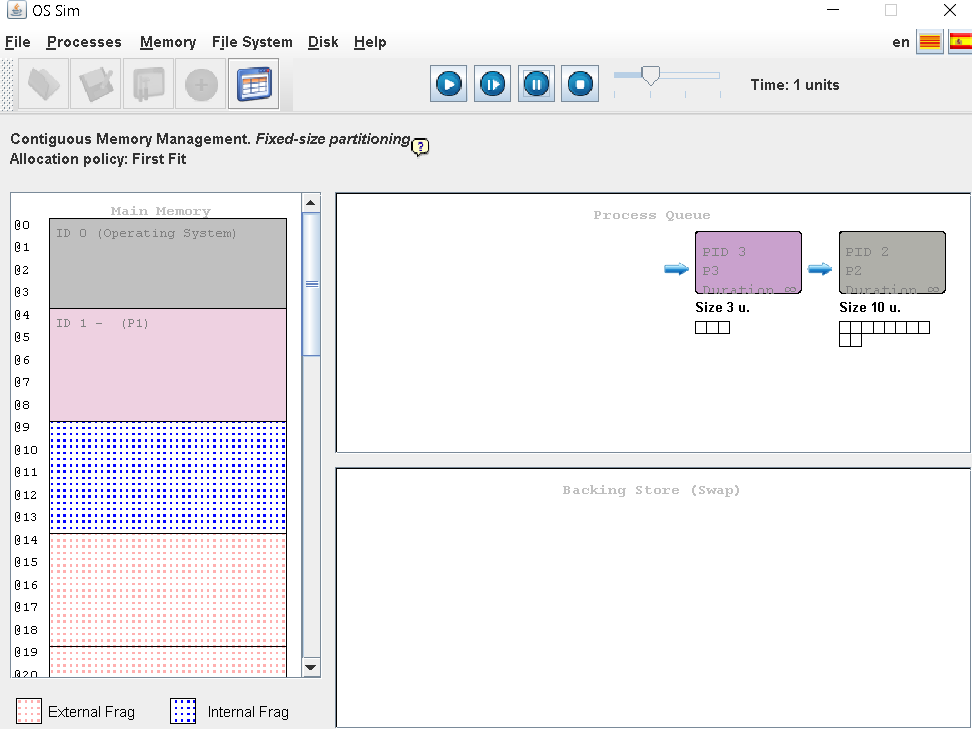
* 1. **Contiguous memory management dengan menggunakan partisi berukuran tetap (ficed-size partition) dan aturan first fit.**

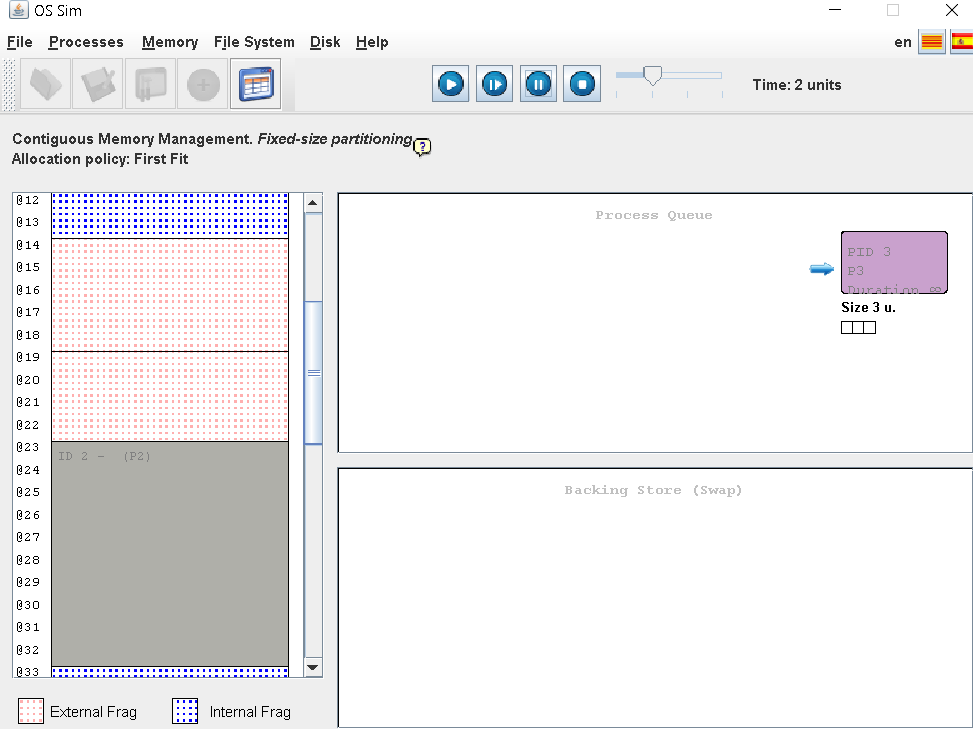
Memori dibagi menjadi partisi-partisi berukuran tetap, lalu seluruh proses akan dialokasikan pada satu dari partisi yang tersedia. Pada aturan first fit, partisi pertama yang dapat memuat proses akan dipilih.

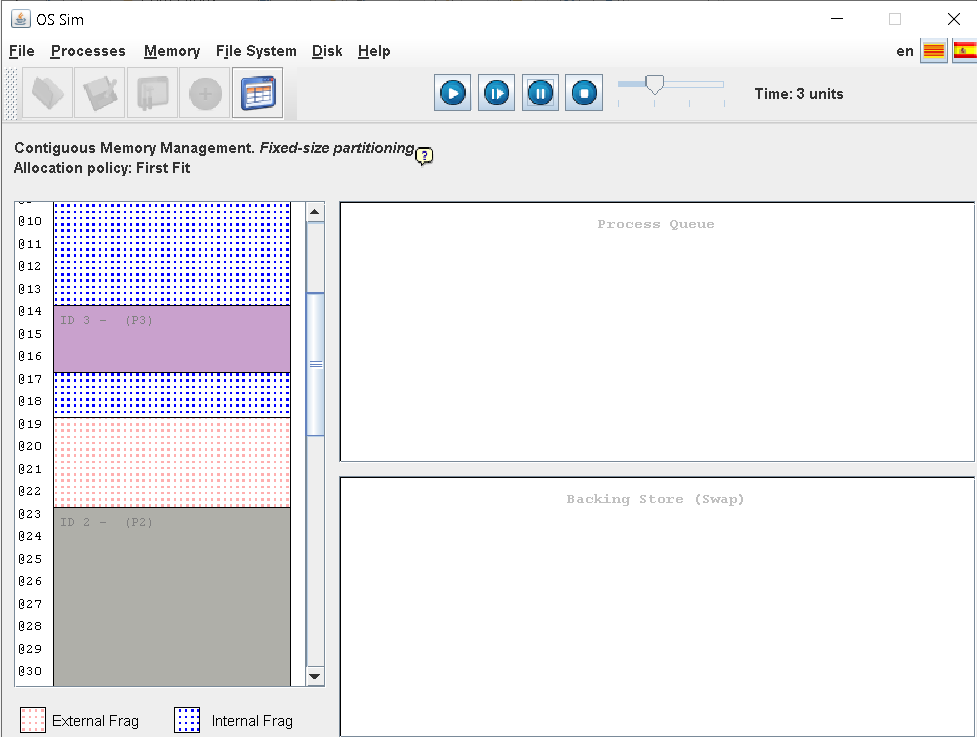
1. Pilih menu help >> examples...>> memory management >> Contiguous memory management with fixed-size partitions (first adjustment). Pilihlah folder hingga menampilkan sebagai berikut.



1. Selanjutnya mengeklik tombol play untuk melihat proses yang terjadi dan mengeklik tombol stop untuk berhenti.
2. Melakukan Analisa proses dengan melakukan klik pada tombol next.







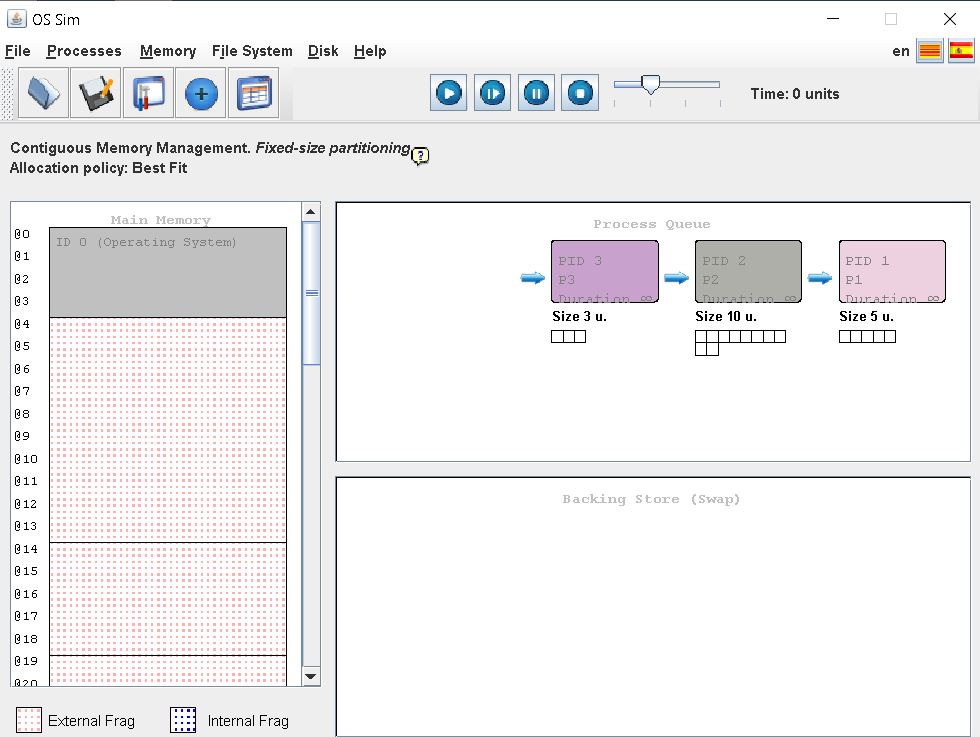
Dengan aturan first fit, banyak terjadi internal fragmentasi. Ini terjadi karena ketika suatu proses menemukan partisi yang cukup untuk dimuat, makai a akan dimuat di partisi tersebut tanpa memandang apakah sisa partisi banyak atau tidak.

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

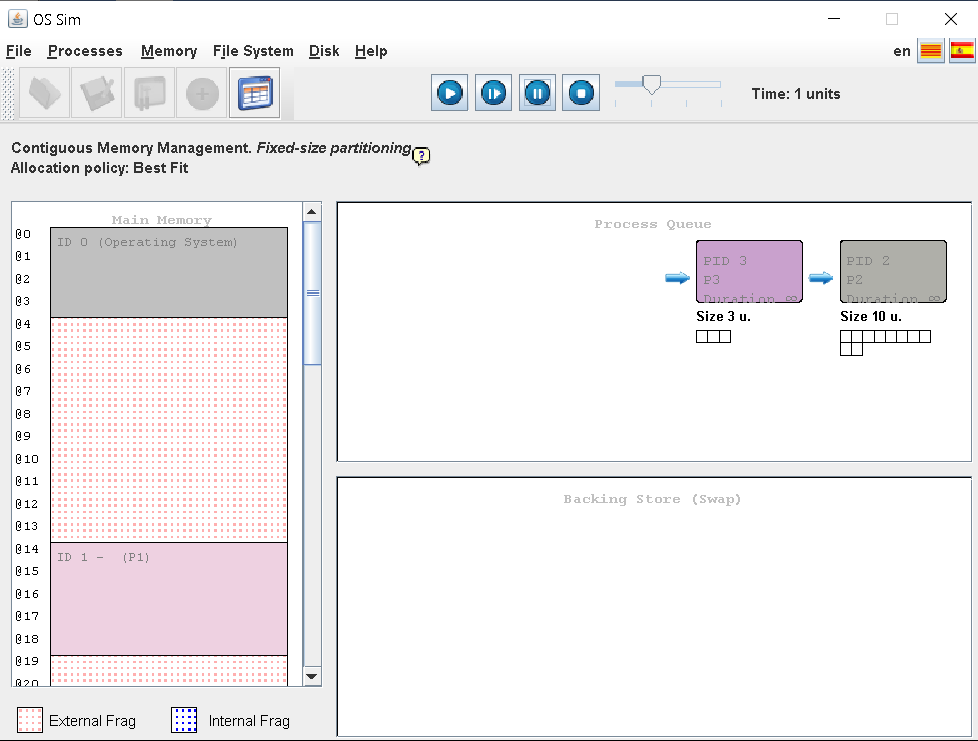
Memori dibagi menjadi partisi-partisi berukuran tetap, lalu seluruh proses akan dialokasikan pada satu dari partisi yang tersedia. Pada aturan best fit, partisi terbaik (yang ukurannya sama atau hampir sama) yang dapat memuat proses akan dipilih

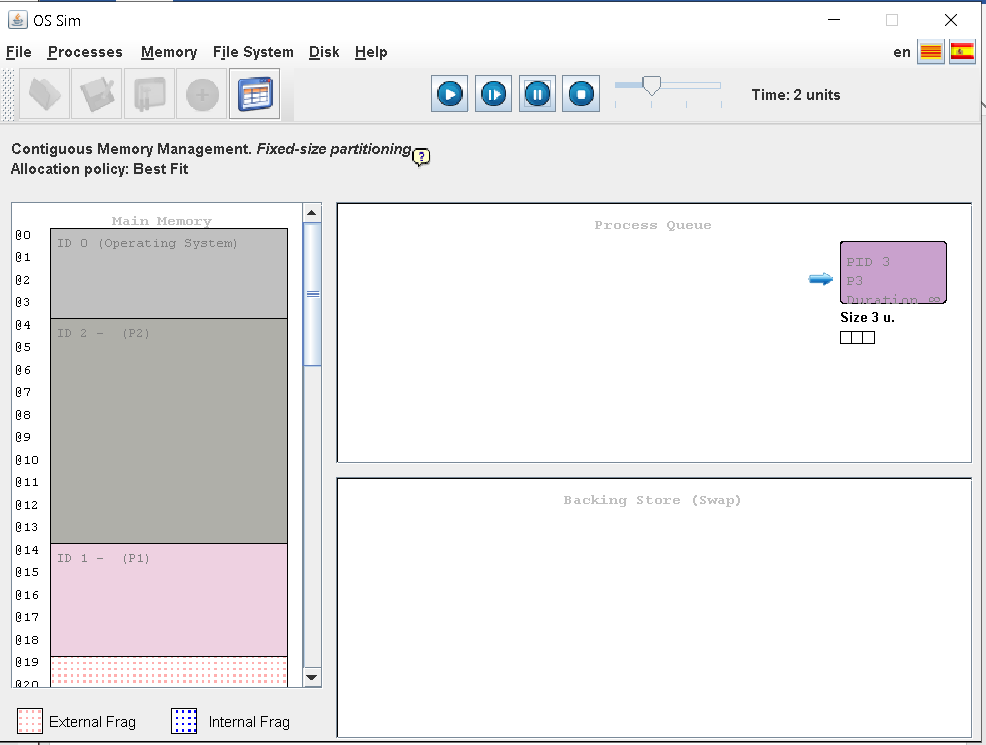
Langkah:

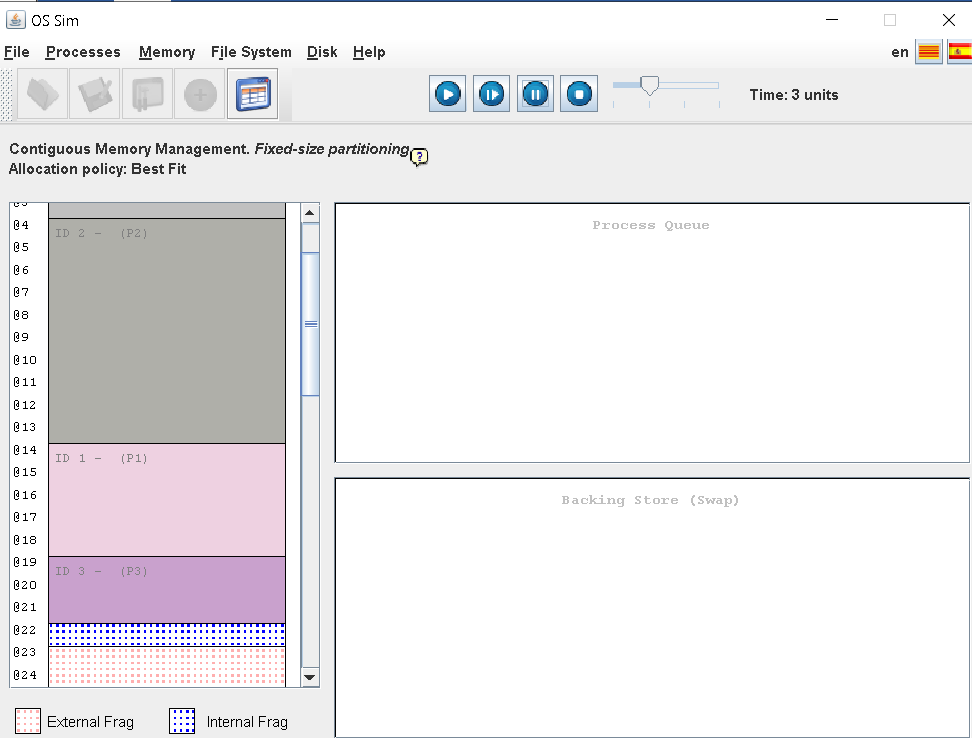
* + 1. Memilih menu help >> examples..>> memory management >> Contiguous memory management with fixed-size partitions (best fit)>> pilih folder



* + 1. Selanjutnya mengeklik tombol play untuk melihat proses yang terjadi dan mengeklik tombol stop untuk berhenti.
    2. Melakukan Analisa proses dengan mengeklik pada tombol next.







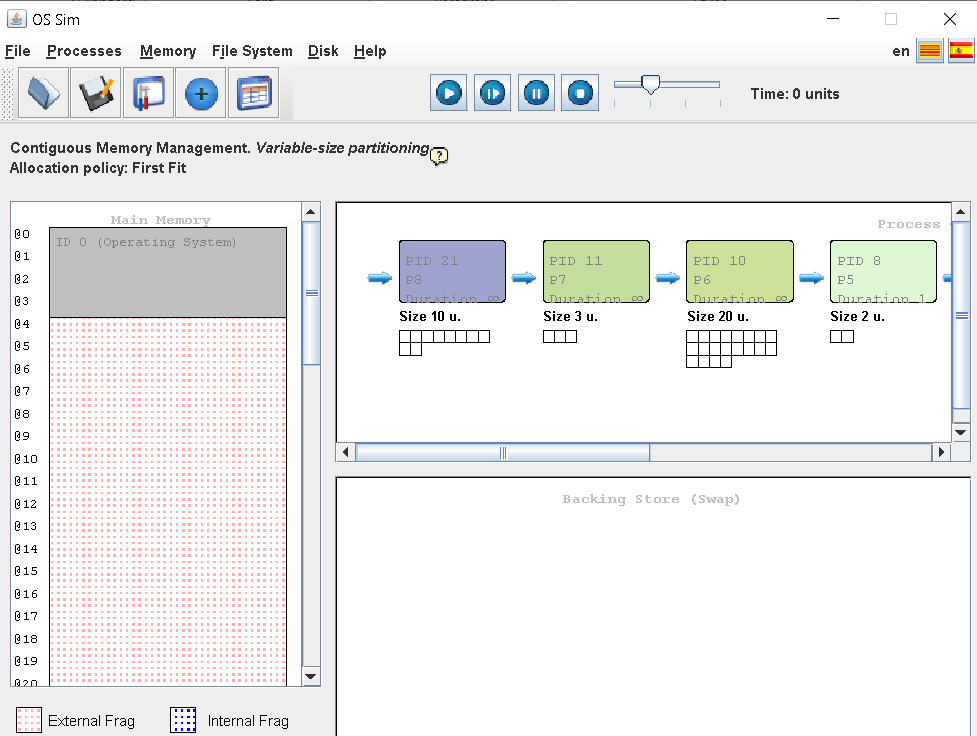
Dengan menggunakan aturan best fit, internal fragmentasi berkurang. Ini terjadi karena proses dimuat pada ukuran partisi terkecil akan tetapi masih dapat memuat proses tersebut sehingga internal fragmentasi dapat diminimalisasi.

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

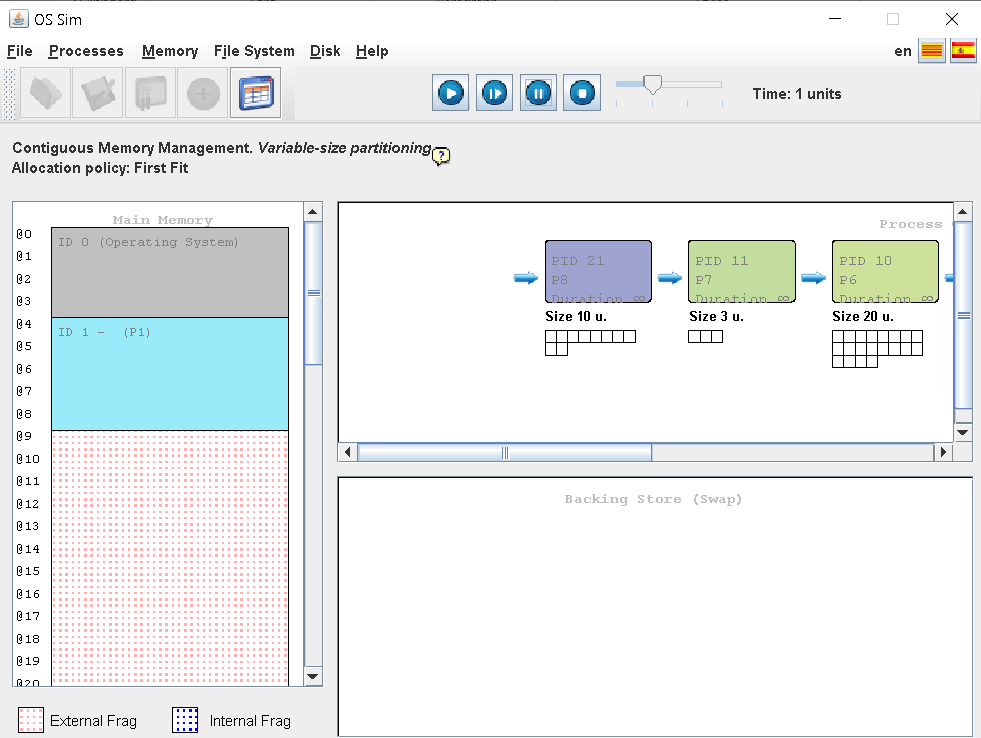
Memori pertama tidak dipartisi, lalu partisi akan dibuat saat proses dialokasikan dan mempunyai ukuran yang sama dengan ukuran proses. Selanjutnya partisi akan dibebaskan (saat proses sudah selesai menggunakan memori) dan digunakan oleh proses lain. Jika ukuran proses lebih kecil dari ukuran partisi,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

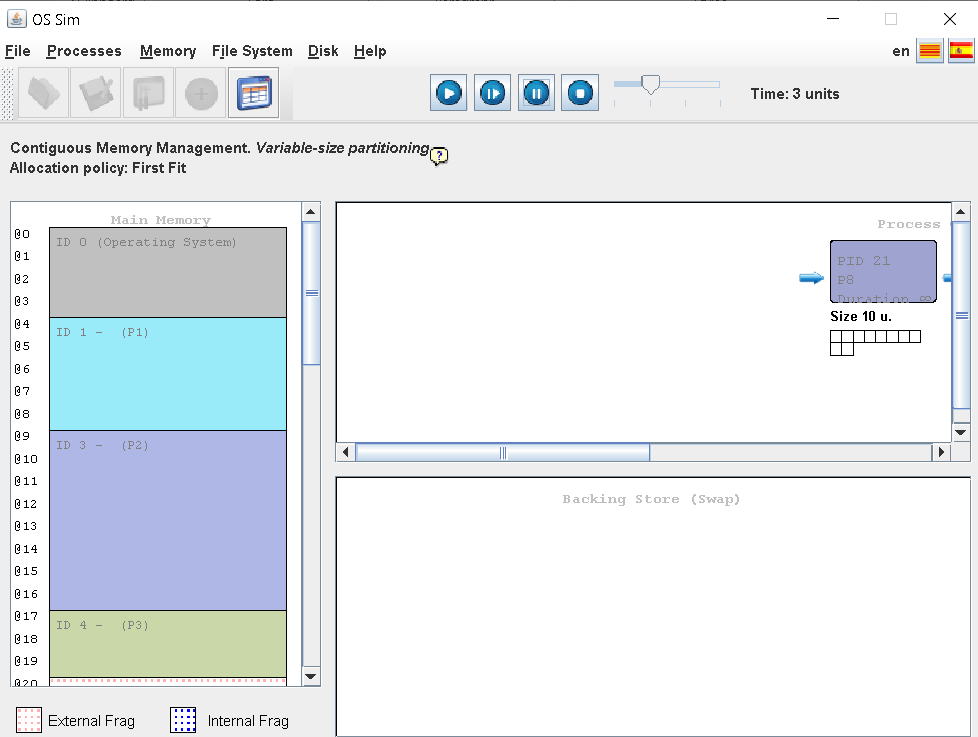
Langkah:

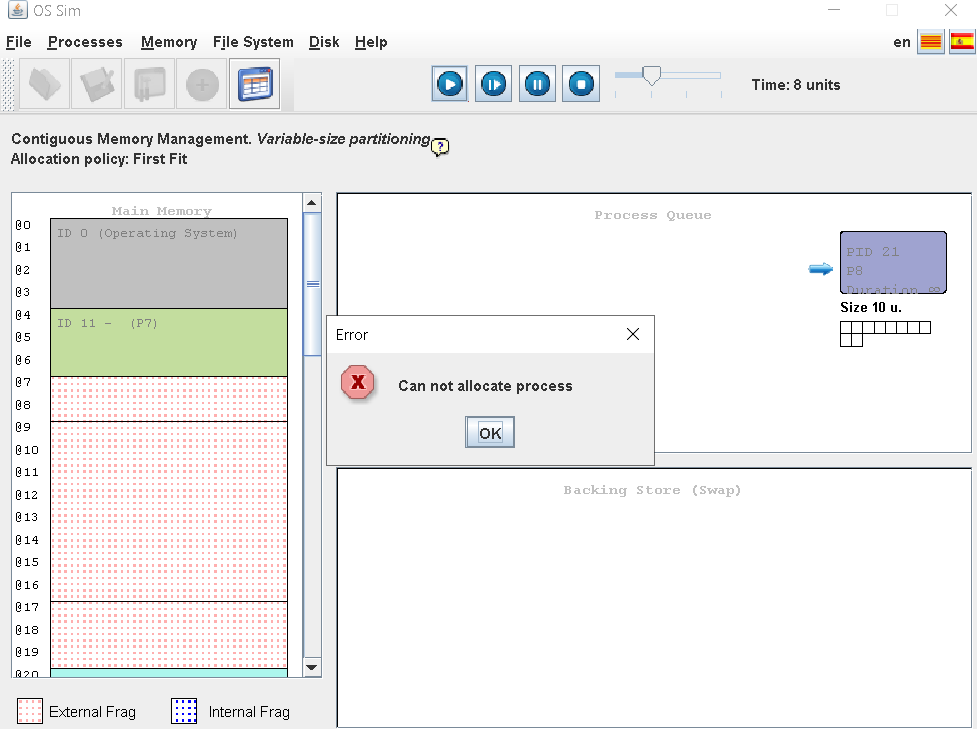
* + 1. Memilih menu help >> examples...>> memory management >>Contiguous memory management with variable-sized partitions (Defragmentation) >> pilih folder



* + 1. Selanjutnya mengeklik tombol play untuk melihat proses yang terjadi dan mengeklik tombol stop untuk berhenti .
    2. Melakukan analisa proses dengan melakukan klik pada tombol next.







Pada langkah terakhir terdapat sebuah proses yang tidak dapat dimuat. Ini terjadi karena pembuatan partisi berdasarkan proses yang dimuat pertama kali tetapi tidak ada partisi yang dapat memuat proses terakhir tersebut

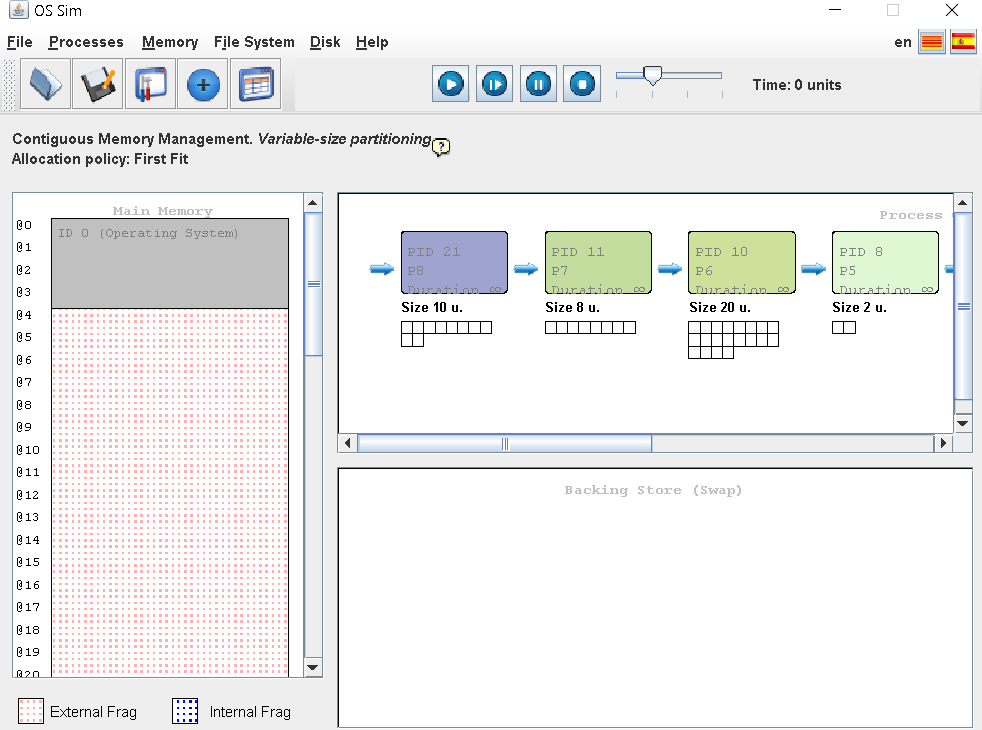
# Contiguous memory management dengan menggunakan partisi berukuran tidak tetap (variable-size partition) >> swap

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.

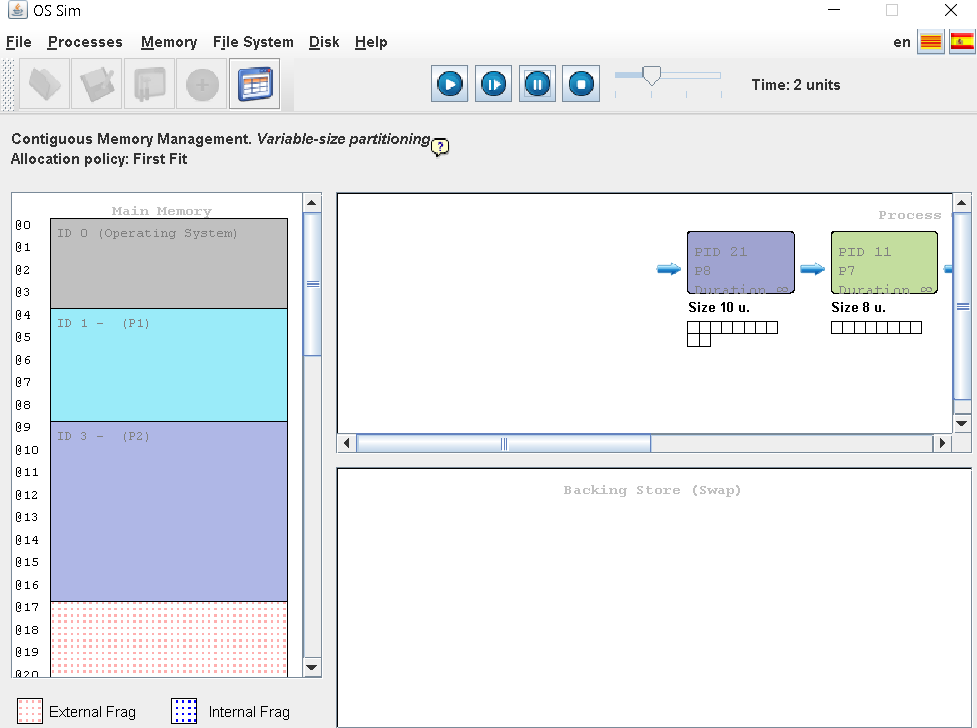
Memori mempunyai ukuran yang berhingga, hanya beberapa proses saja yang bisa dimuat di memori pada saat yang sama, tetapi space swap memungkinkan kita untuk menyimpan beberapa proses yang sedang kurang aktif, sehingga space pada memori dapat dialokasikan untuk proses yang lain yang lebih aktif. Proses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

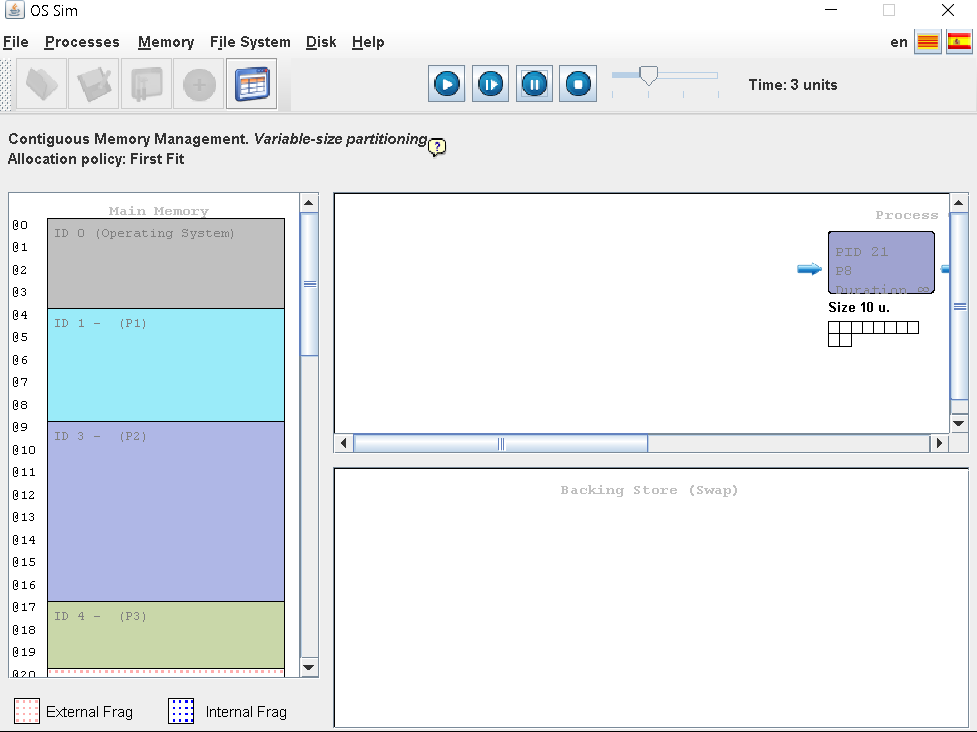
Langkah:

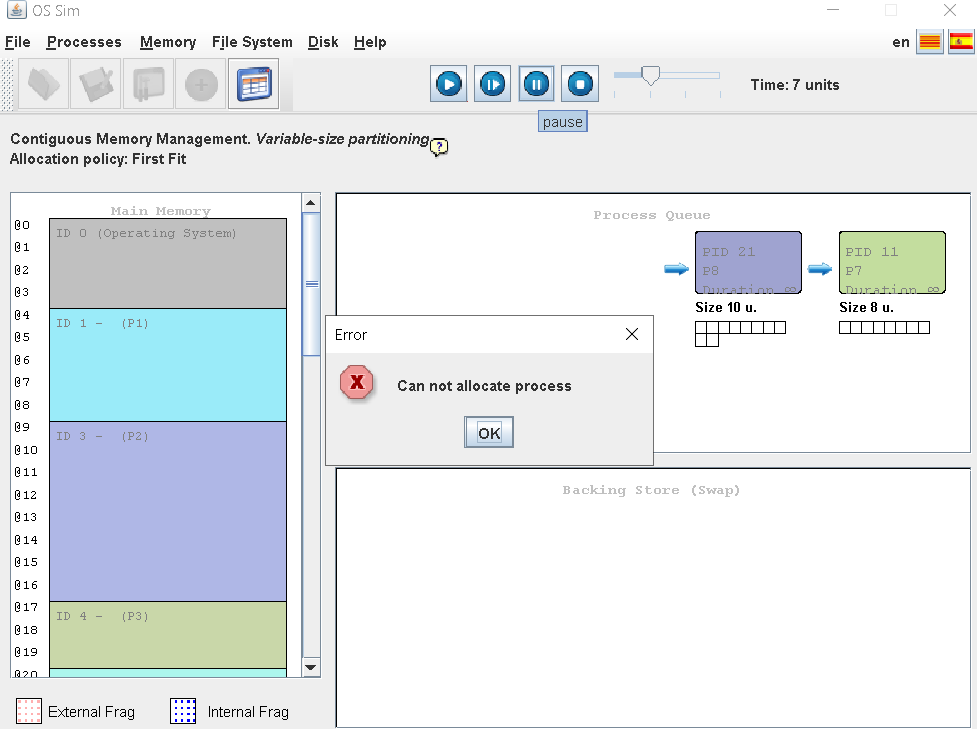
1. Memilih menu help >> examples...>> memory management >>Contiguous memory management with variable-sized partitions (Swap) >> pilih folder



1. Selanjutnya mengeklik tombol play untuk melihat proses yang terjadi dan mengeklik tombol stop untuk berhenti.
2. Melakukan Analisa proses dengan melakukan klik pada tombol next.







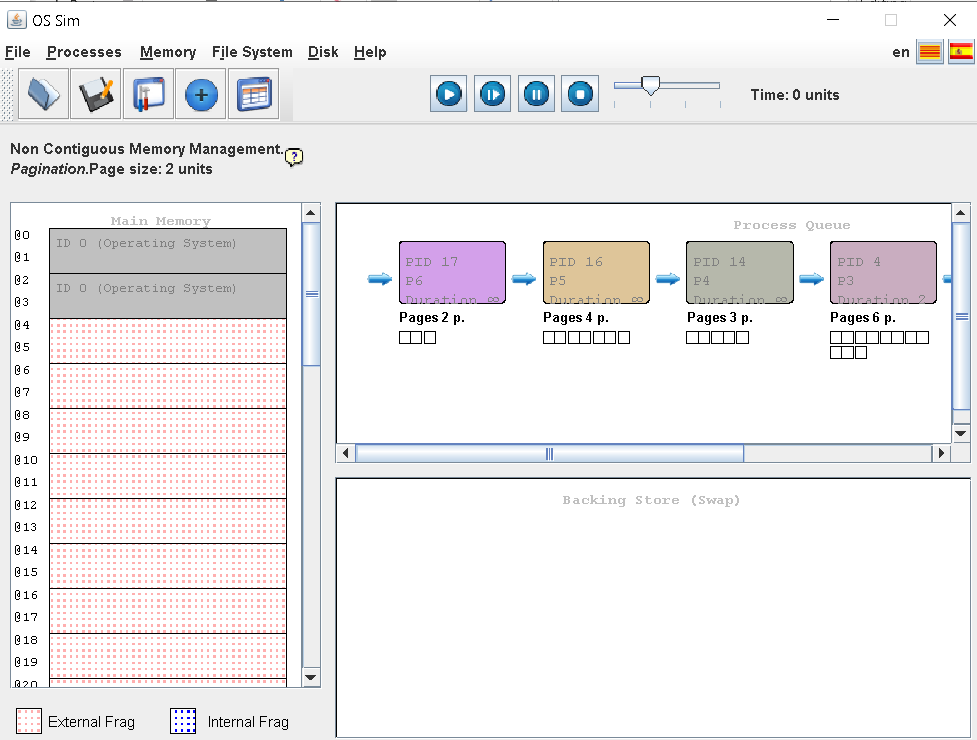
Sampai langkah ke 7 akan ada notifikasi Can not allocate process. Hal ini mungkin terjadi karena tidak adanya proses yang tidak aktif sehingg tidak terjadi swap dan partisi tidakada yang mencukupi untuk memuat proses yang tersisa.

# Pagination (ukuran page 2 unit)

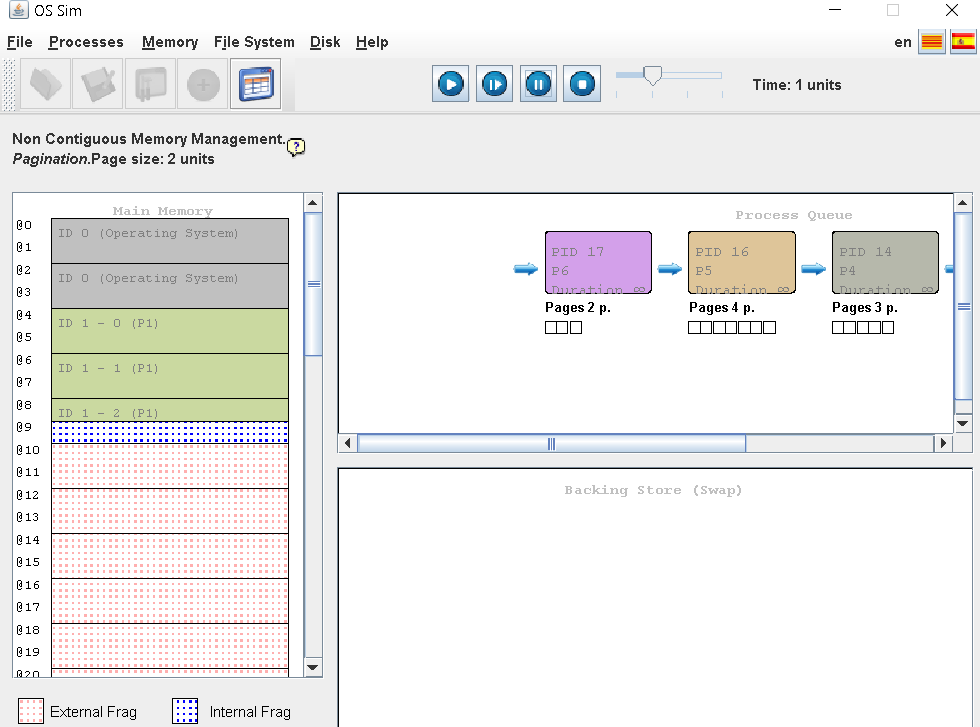
Proses akan dibagi menjadi beberapa page dengan ukuran tetap (contohnya adalah 2 unit), memori lalu dibagi menjadi beberapa frame berukuran sama. Page dari proses lalu dialokasikan pada frame memori yang kosong.

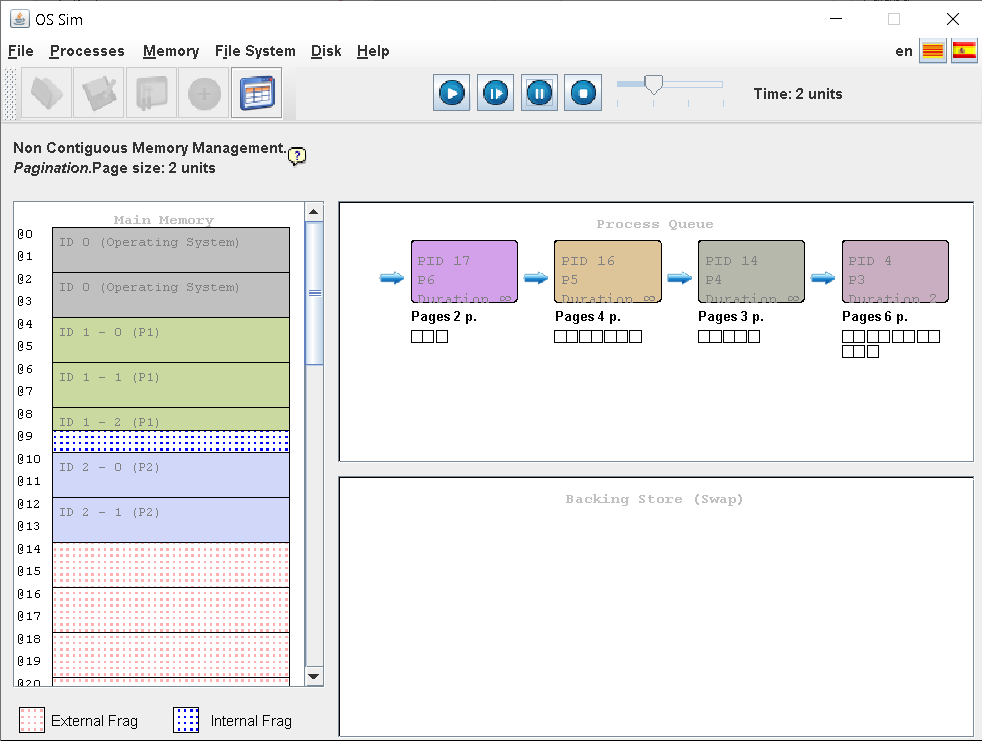
Langkah:

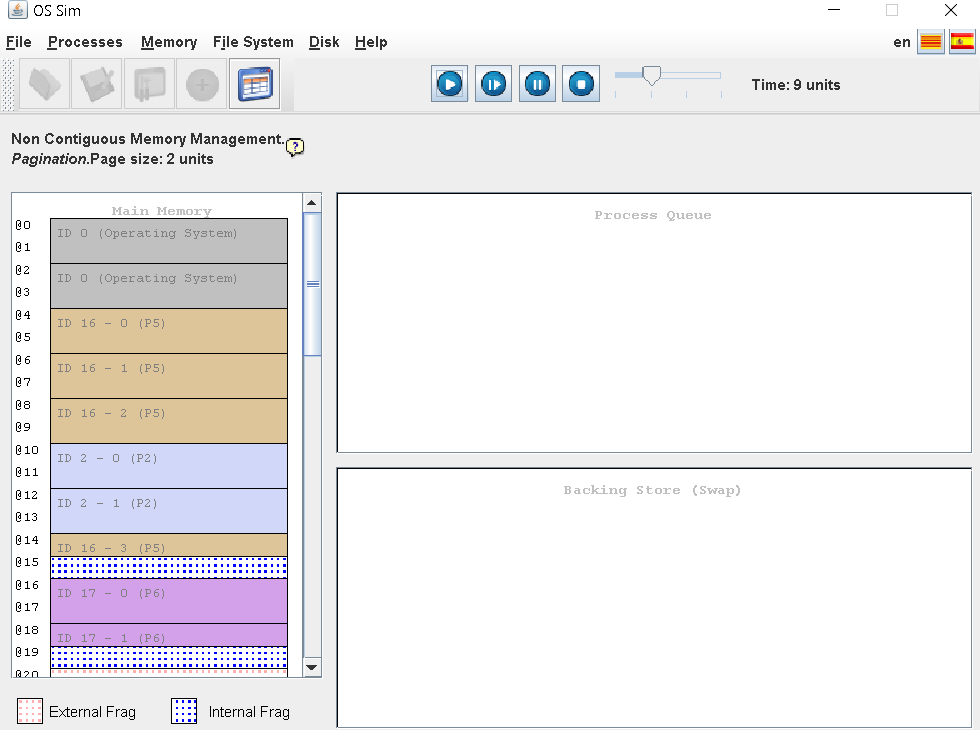
* + 1. Memilih menu help >> examples...>> memory management >>Pagination) >> pilih folder



* + 1. Selanjutnya mengeklik tombol play untuk melihat proses yang terjadi dan mengeklik tombol stop untuk berhenti .
    2. Melakukan analisa proses dengan melakukan klik pada tombol next.







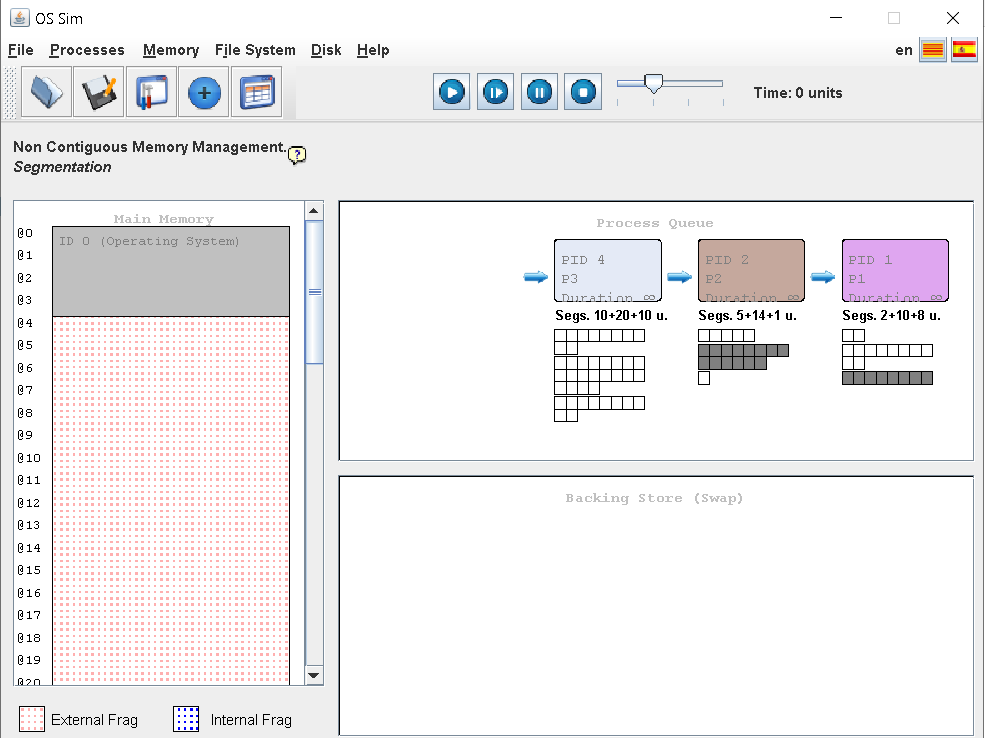
Terlihat pada percobaan ini bahwa setiap proses dibagi menjadi beberapa page denganukuran tetap setiap pagenya, yaitu 2 unit.

# Segmentation (alokasi parsial)

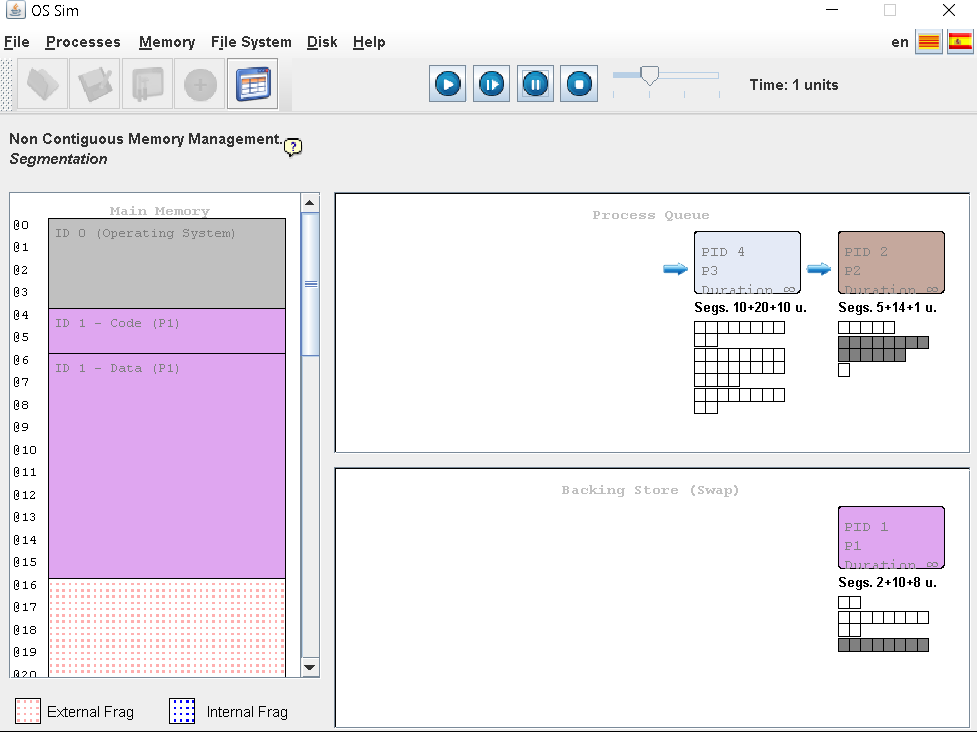
Proses dibagi menjadi beberapa segmen, tiap segmen bisa mempunyai ukuran yang berbeda. Segmen ini akan dialokasikan ke memori secara independen, lalu partisi akan dibuat untuk menampung segmen tersebut dengan ukuran yang sama. Pembagian ini adalah secara fungsional dan berdasarkan pada logical structure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

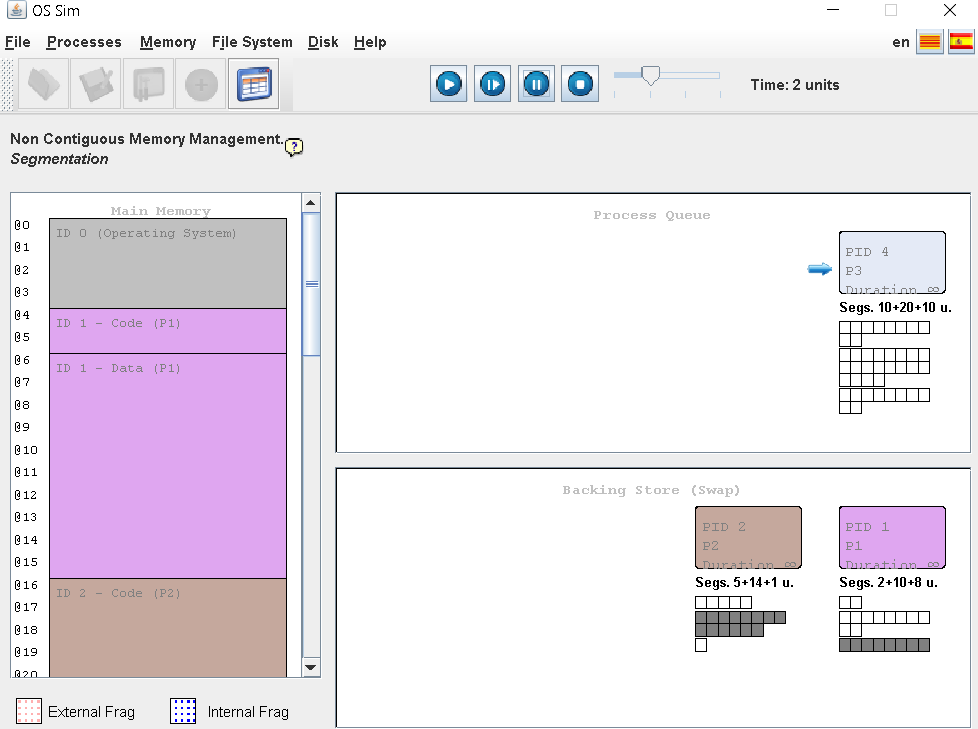
Langkah:

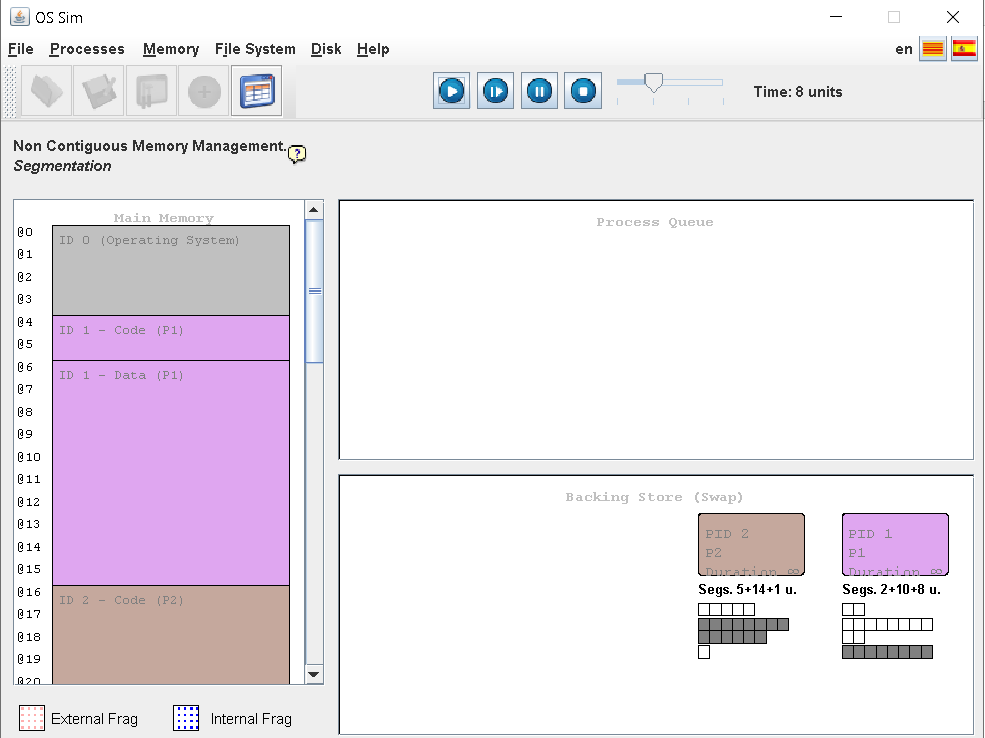
* + 1. Memilih menu help >> examples...>> memory management >>Segmentasi >> pilih folder



* + 1. Selanjutnya mengeklik tombol play untuk melihat proses yang terjadi dan mengeklik tombol stop untuk berhenti.
    2. Lakukan analisa proses dengan melakukan klik pada tombol next.







Semua proses dapat dimuat, karena setiap proses dibagi menjadi beberapa segmen berdasarkan logical structure-nya. Dan karena tidak semua segmen proses tersebut dimuat di memori, semua proses dapat dimuat. Dan yang segmen yang tidak dimuat tadi dapat di swap-out