Clase Práctica - Scheduling

Sistemas Operativos
DC - UBA - FCEN

29 de Marzo de 2016

Repaso de Conceptos de la Teórica

- **Proceso**:
  Instancia de un programa en ejecución.

Repaso de Conceptos de la Teórica

- **Proceso**:
  Instancia de un programa en ejecución. Proceso = Programa (código) + memoria (variables) + registros (¡PC!).

Repaso de Conceptos de la Teórica

- **Proceso**:
  Instancia de un programa en ejecución. Proceso = Programa (código) + memoria (variables) + registros (¡PC!).
- **Sistema operativo multitarea/multiprogramado**:
  Sistema que puede contener muchos procesos compartiendo recursos, donde el más importante suele ser el CPU.

Repaso de Conceptos de la Teórica

- **Proceso**:
  Instancia de un programa en ejecución. Proceso = Programa (código) + memoria (variables) + registros (¡PC!).
- **Sistema operativo multitarea/multiprogramado**:
  Sistema que puede contener muchos procesos compartiendo recursos, donde el más importante suele ser el CPU.
- **Scheduler**:
  Módulo del sistema operativo que se encarga de la remoción, selección y reemplazo del proceso en ejecución.

Repaso de Conceptos de la Teórica

- **Proceso**:
  Instancia de un programa en ejecución. Proceso = Programa (código) + memoria (variables) + registros (¡PC!).
- **Sistema operativo multitarea/multiprogramado**:
  Sistema que puede contener muchos procesos compartiendo recursos, donde el más importante suele ser el CPU.
- **Scheduler**:
  Módulo del sistema operativo que se encarga de la remoción, selección y reemplazo del proceso en ejecución.
- **Context-switch**:
  Procedimiento mediante el cual el sistema cambia el proceso en ejecución. Involucra guardar el estado del proceso que es desalojado (si no terminó) y cargar el estado del proceso que va a ejecutarse a continuación.

Diagrama de Estados

- A medida que un proceso se ejecuta va cambiando de estado de acuerdo a su actividad y a las decisiones tomadas por el SO.
- El **diagrama de estados** muestra los distintos estados atravesados por los procesos en un sistema y cómo un proceso pasa de un estado a otro.

Diagrama de Estados

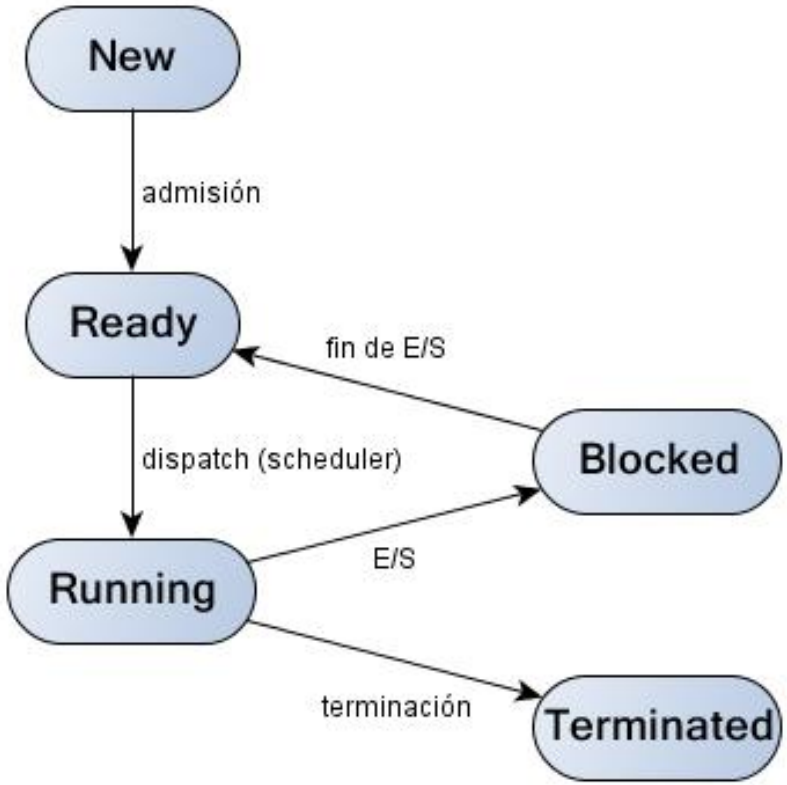
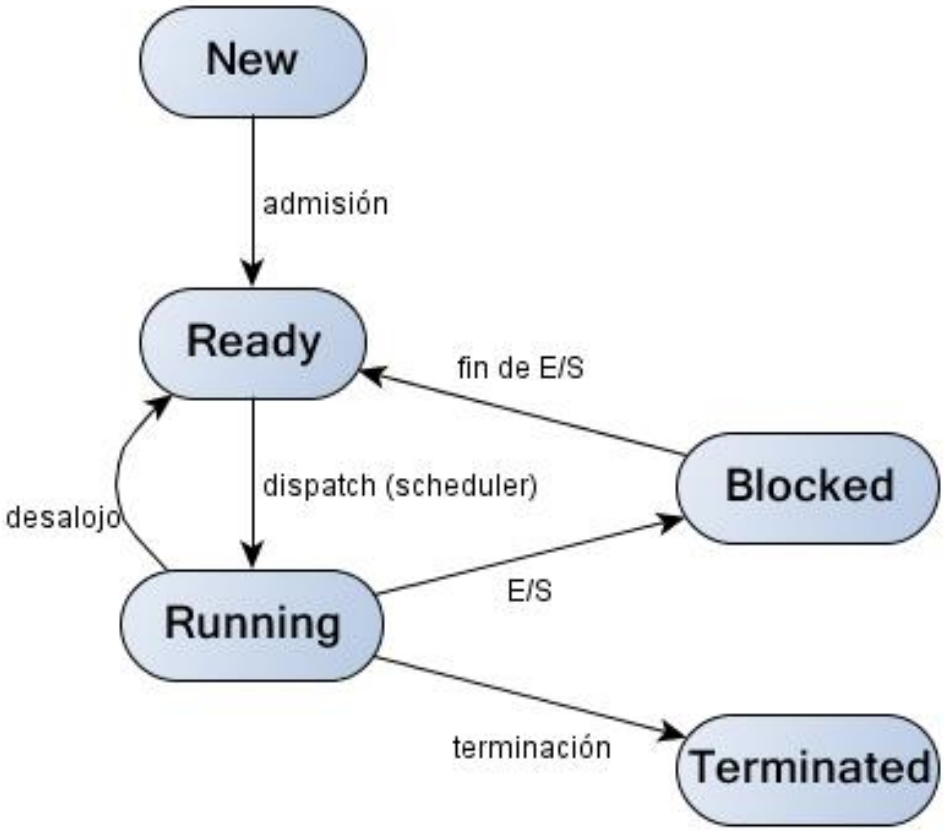


Diagrama de Estados

¿Y si permite desalojo?

Diagrama de Estados

¿Y si permite desalojo?



Políticas de Scheduling

¿Cómo elegimos el siguiente proceso a ejecutar?

- **FCFS** (First Came, First Served)
  La CPU se asigna a los procesos en el orden en el que la solicitan.

Políticas de Scheduling

¿Cómo elegimos el siguiente proceso a ejecutar?

- **FCFS** (First Came, First Served)
  La CPU se asigna a los procesos en el orden en el que la solicitan.
- **Prioridades fijas**
  Cada proceso tiene un valor de prioridad asignado y se ejecutan primero los de mayor prioridad.

Políticas de Scheduling

¿Cómo elegimos el siguiente proceso a ejecutar?

- **FCFS** (First Came, First Served)
  La CPU se asigna a los procesos en el orden en el que la solicitan.
- **Prioridades fijas**
  Cada proceso tiene un valor de prioridad asignado y se ejecutan primero los de mayor prioridad.
- **SJF** (Shortest Job First)
  Primero se ejecutan los procesos de menor duración. Para esto debe conocerse la duración de antemano.

Políticas de Scheduling

¿Cómo elegimos el siguiente proceso a ejecutar?

- **SRTF** (Shortest Remaining Time First)
  Primero se ejecutan los procesos a los que les resta menos tiempo de CPU. También debe conocerse la duración y llevarla cuenta del tiempo ejecutado.

Políticas de Scheduling

¿Cómo elegimos el siguiente proceso a ejecutar?

- **SRTF** (Shortest Remaining Time First)
  Primero se ejecutan los procesos a los que les resta menos tiempo de CPU. También debe conocerse la duración y llevarla cuenta del tiempo ejecutado.
- **Round Robin**
  La idea es darle un quantum a cada proceso, e ir alternando entre ellos.

Políticas de Scheduling

¿Cómo elegimos el siguiente proceso a ejecutar?

- **SRTF** (Shortest Remaining Time First)
  Primero se ejecutan los procesos a los que les resta menos tiempo de CPU. También debe conocerse la duración y llevarla cuenta del tiempo ejecutado.
- **Round Robin**
  La idea es darle un quantum a cada proceso, e ir alternando entre ellos.
- **Múltiples colas**
  Existen varias colas de procesos en estado "Ready", con distintas prioridades. Los procesos se asignan a una cola, generalmente en función de alguna propiedad del proceso.

Ejercicio 1

Dados los siguientes procesos,

Proceso	Tiempo de llegada	Tiempo de ejecución	Tiempo de bloqueo (inc.)	Prioridad
P0	0	6	2-4	3
P1	1	3	-	2
P2	2	6	-	1
P3	3	4	-	4

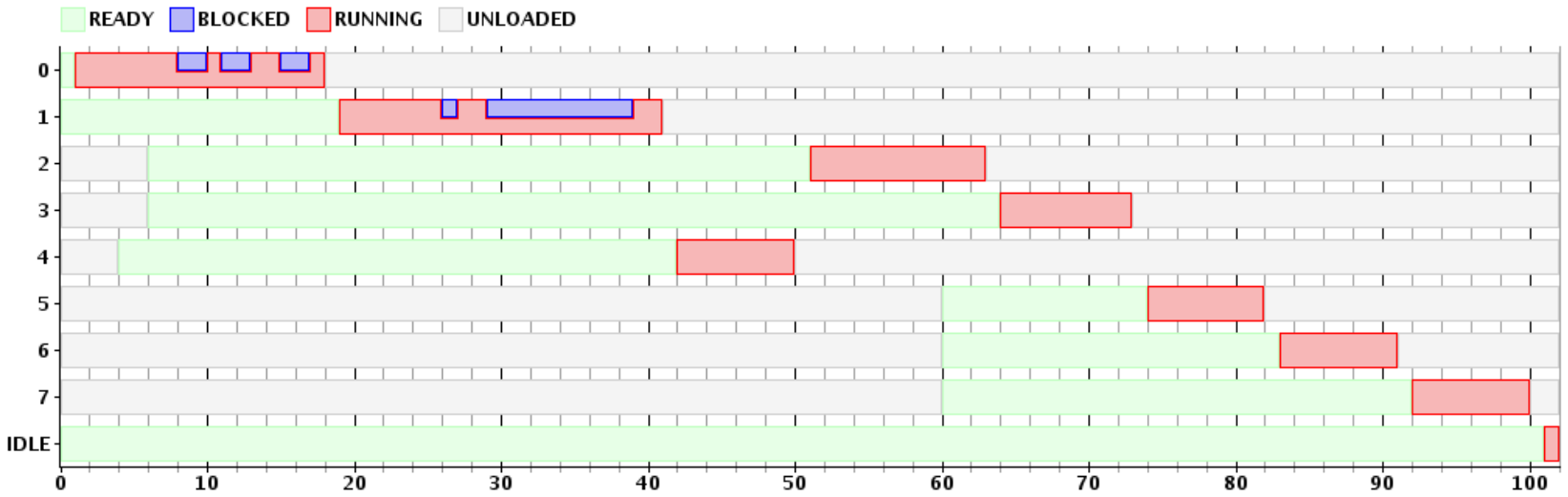
Costo de cambio de contexto = 1 unidad de tiempo
Costo de *load* de un proceso = 1 unidad de tiempo.

Dibujar un diagrama de **GANTT** para cada una de las siguientes políticas de *scheduling* y calcular cuál es la mejor:

- FCFS
- Prioridades (Sin desalojo)
- Round Robin con quantum = 2 unidades de tiempo

Diagrama de GANTT

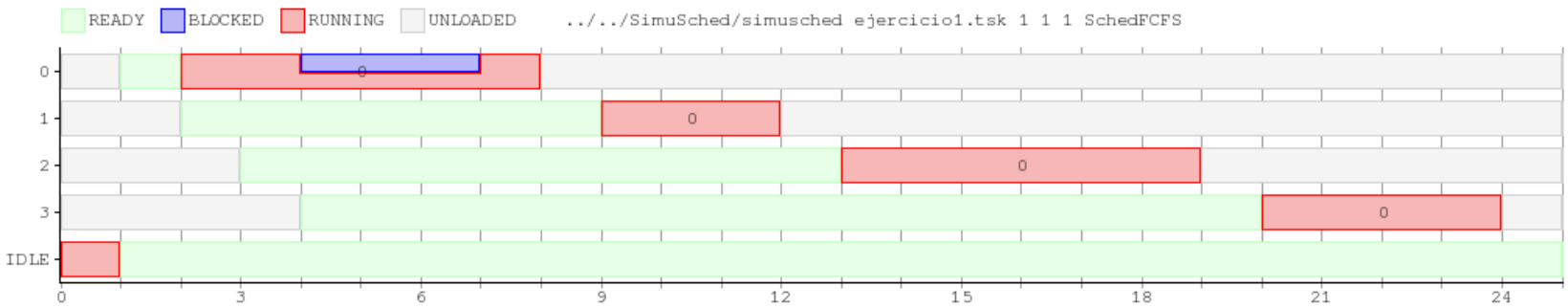
- Un diagrama de **GANTT** es una herramienta gráfica cuyo objetivo es mostrar el **estado** de cada uno de los **procesos** existentes en un sistema durante un período de tiempo determinado.



Solución

Diagramas de Gantt:

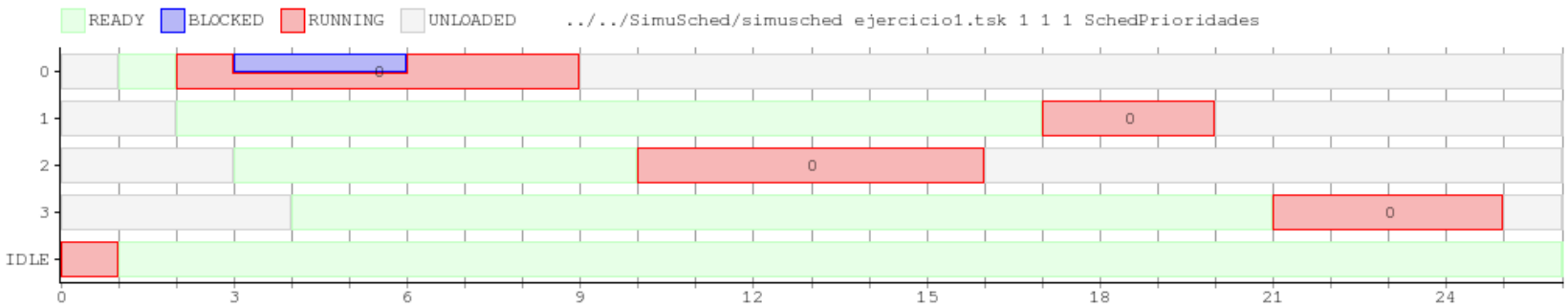
- FCFS:



Solución

Diagramas de Gantt:

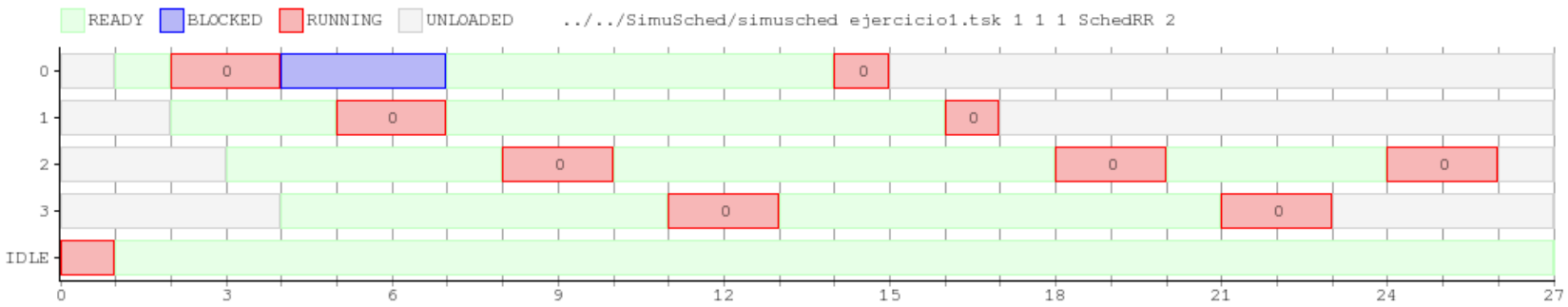
- Prioridades:



Solución

Diagramas de Gantt:

- Round Robin:



Rendimiento

¿Cómo evaluamos el rendimiento de los distintos algoritmos de *scheduling*?
Hay distintas métricas (no siempre compatibles entre ellas)

- *Fairness*:
  "Justicia" en la asignación del CPU.

Rendimiento

¿Cómo evaluamos el rendimiento de los distintos algoritmos de *scheduling*?
Hay distintas métricas (no siempre compatibles entre ellas)

- *Fairness*:
  "Justicia" en la asignación del CPU.
- Tiempo de respuesta:
  Tiempo que el proceso tarda en empezar a ejecutarse.

Rendimiento

¿Cómo evaluamos el rendimiento de los distintos algoritmos de *scheduling*?
Hay distintas métricas (no siempre compatibles entre ellas)

- *Fairness*:
  "Justicia" en la asignación del CPU.
- Tiempo de respuesta:
  Tiempo que el proceso tarda en empezar a ejecutarse.
- *Throughput*:
  Cantidad de procesos que terminan por unidad de tiempo.

Rendimiento

¿Cómo evaluamos el rendimiento de los distintos algoritmos de *scheduling*?
Hay distintas métricas (no siempre compatibles entre ellas)

- *Fairness*:
  "Justicia" en la asignación del CPU.
- Tiempo de respuesta:
  Tiempo que el proceso tarda en empezar a ejecutarse.
- *Throughput*:
  Cantidad de procesos que terminan por unidad de tiempo.
- *Turnaround*:
  Tiempo total que le toma a un proceso ejecutar completamente.

Rendimiento

¿Cómo evaluamos el rendimiento de los distintos algoritmos de *scheduling*?
Hay distintas métricas (no siempre compatibles entre ellas)

- *Fairness*:
  "Justicia" en la asignación del CPU.
- Tiempo de respuesta:
  Tiempo que el proceso tarda en empezar a ejecutarse.
- *Throughput*:
  Cantidad de procesos que terminan por unidad de tiempo.
- *Turnaround*:
  Tiempo total que le toma a un proceso ejecutar completamente.
- *Waiting time*:
  Tiempo que un proceso pasa en estado `ready`.

Ejercicio 2

Dados los siguientes procesos,

Proceso	Tiempo de llegada	Tiempo de ejecución	Tiempo de bloqueo (inc)	Prioridad
P0	0	6	2-4	3
P1	1	3	-	2
P2	2	6	-	1
P3	3	4	-	4

- Calcular el *waiting time* promedio de cada una de las siguientes políticas de *scheduling*:
  - FCFS
  - Piroridades fijas
  - Round robin

Ejercicio 2

Waiting time promedio con cada algoritmo:

$$\begin{array}{lcccccc} \text{FCFS} & = & \frac{1+7+10+16}{4} & = & \frac{34}{4} & = & 8.5 \\ \text{Prioridades} & = & \frac{1+15+7+17}{4} & = & \frac{40}{4} & = & 10 \\ \text{Round Robin} & = & \frac{(1+7)+(3+9)+(5+8+4)+(7+8)}{4} & = & \frac{42}{4} & = & 13 \end{array}$$

- ¿En qué situación elegiría Round-Robin en lugar de alguno de los otros aunque su *waiting time* promedio sea mayor?

Ejercicio 2

Waiting time promedio con cada algoritmo:

$$\begin{array}{lcccccc} \text{FCFS} & = & \frac{1+7+10+16}{4} & = & \frac{34}{4} & = & 8.5 \\ \text{Prioridades} & = & \frac{1+15+7+17}{4} & = & \frac{40}{4} & = & 10 \\ \text{Round Robin} & = & \frac{(1+7)+(3+9)+(5+8+4)+(7+8)}{4} & = & \frac{42}{4} & = & 13 \end{array}$$

- ¿En qué situación elegiría Round-Robin en lugar de alguno de los otros aunque su *waiting time* promedio sea mayor?
- Analicemos inanición para las políticas de *scheduling* anteriores.

Ejercicio 3: ¿Y si tenemos un sistema con 2 núcleos?

Dados los siguientes procesos,

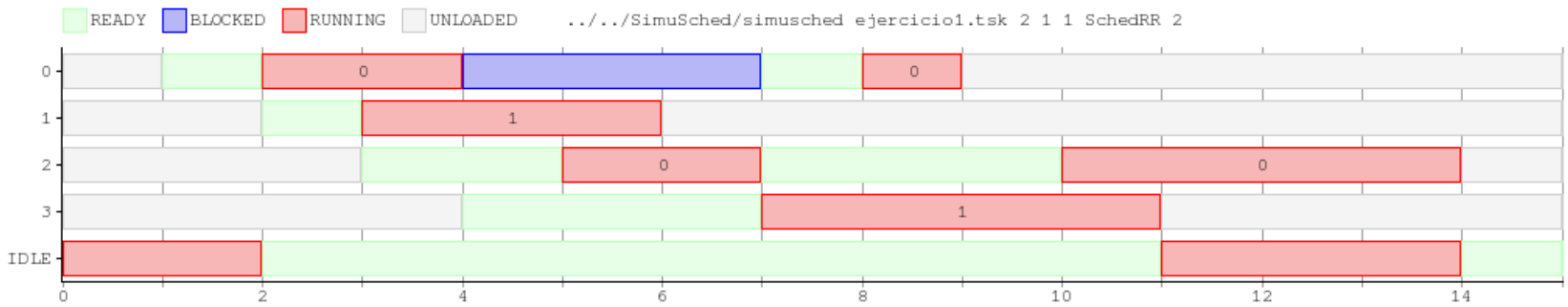
Proceso	Tiempo de llegada	Tiempo de ejecución	Tiempo de bloqueo (incl)	Prioridad
P0	0	6	2-4	3
P1	1	3	-	2
P2	2	6	-	1
P3	3	4	-	4

- *context switch* = 1 unidad de tiempo
- *load* = 1 unidad de tiempo.
- Migración entre núcleos = 2 unidades de tiempo.

- Dibujar el diagrama de Gantt para la política de *scheduling*:
  - Round Robin con quantum = 2 unidades de tiempo, permitiendo la migración entre núcleos.

Solución

- Diagrama de Gantt:



¿Preguntas?