Parte 1. Arquitectura del Sistema

Queremos construir un programa multihilo (`pthread.h`) que simule el funcionamiento del kernel de un sistema operativo. En esta primera parte sólo habrá que definir el marco de funcionamiento; y en las siguientes partes, se irán ampliando las capacidades del sistema.

La programación del sistema deberá ser realizada utilizando el lenguaje de programación C. El simulador deberá:

- Recibir como parámetros las opciones de configuración.
- Inicializar las estructuras de datos que se vayan a utilizar.
- Lanzar los hilos que implementarán los distintos subsistemas.
- Realizar la comunicación utilizando memoria compartida y los elementos de sincronización vistos en clase.

Los distintos elementos iniciales del sistema que hay que implementar en esta primera parte se encuentran representados en la Figura 1 y descritos a continuación.

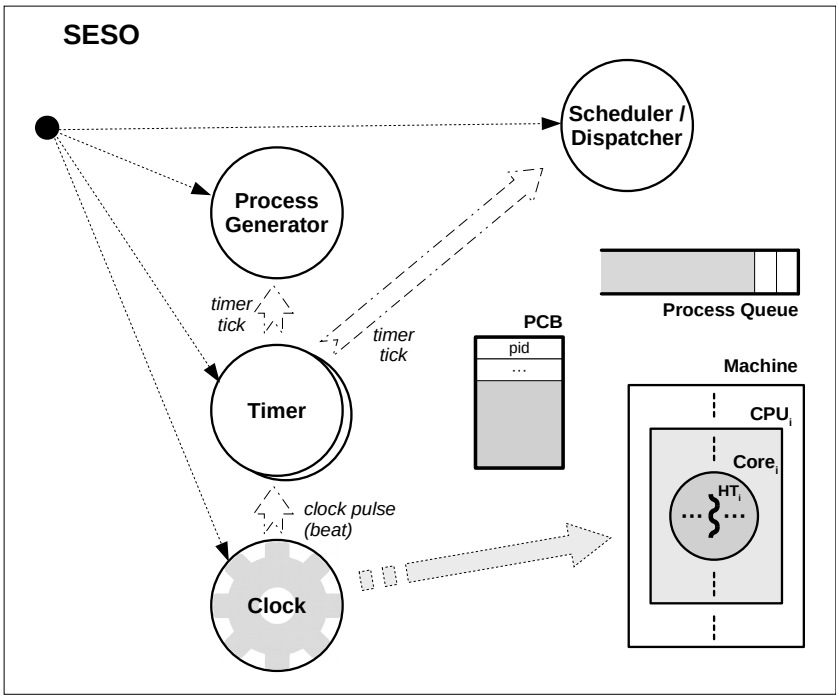

Figura 1. Esquema de los elementos del sistema.

Threads

- **Clock**: Es el motor del simulador: “mueve la máquina (**Machine**)”. Simula el reloj de los procesadores, es decir, cada vez que se ejecuta significa un ciclo de ejecución de cada hilo hardware. Por un lado, deberá hacer avanzar toda la estructura de CPUs, cores e hilos hardware y, por otro, deberá señalar al **Timer** (o *timers*) para que este funcione.
- **Timer**: Se encarga de generar una señal cada cierto periodo de tiempo (parámetro configurable). Recibe los pulsos del **Clock** y con la frecuencia que se haya decidido produce

una interrupción del temporizador, un **tick**.

- **Process Generator**[1]: Genera procesos (**PCB**s) de manera aleatoria con una frecuencia variable (parámetro configurable).
- **Scheduler/Dispatcher**: Se encarga de planificar y de realizar los cambios de contextos de los procesos (en esta primera parte no hará ninguna tarea). Se despertará con cada interrupción del temporizador, es decir, cada **tick** que produzca el **Timer**.

Estructuras de datos

- **PCB**: Estructura de datos que representa a cada proceso. Inicialmente tendrá dos parámetros: un identificador (**pid**) y un tiempo de vida del proceso[2]. Esta estructura de datos irá modificándose según avance la práctica.
- **Process Queue**: Estructura de datos que contiene el conjunto de los procesos (**PCB**s) creados (o existentes).
- **Machine**: Estructura que representa las CPUs, cores e hilos hardware del sistema (parámetro configurable).
-

Funcionamiento

- El **Clock** genera los ciclos que controlan el tiempo en el sistema.
- El **Timer** (o *timers*) se encarga(n) de generar la señal (interrupción) que avisará periódicamente al resto de funciones del sistema (**Scheduler/Dispatcher**, **Process Generator**,…). Cuando estas funciones reciban la señal de momento no harán nada (p.e. pueden lanzar un mensaje que diga que se ha activado).
- El **Process Generator** genera los PCBs de los procesos nuevos. De momento sólo incluirá el identificador del proceso con un tiempo de vida aleatorio.
- La sincronización entre el **Clock** y el **Timer** (o *timers*) y de este (o estos) con el **Scheduler/ Dispatcher** y el **Process Generator** (u otras posibles funciones) será por medio de las primitivas de sincronización: semáforos binarios *mutex* (`pthread_mutex_t`) y variables de condición (`pthread_cond_t`).

Cronología y entregas de la práctica

Las tres partes de la práctica se han planificado de la siguiente forma:

- **Parte 1** (5%): **Arquitectura del Sistema**. Desde el jueves 7 de octubre hasta el miércoles 13 de octubre.
- **Parte 2** (30%): **Planificación**. Desde el miércoles 13 de octubre hasta el viernes 19 de noviembre.
- **Parte 3** (35%): **Gestión de memoria**. Desde el jueves 18 de noviembre hasta el miércoles 22 de diciembre.

Se deberán realizar dos entregas, cada una con el código y el informe correspondiente:

- **Parte 1 + Parte 2** (35%): Viernes 19 de noviembre.
- **Parte 3** (35%): Miércoles 22 de diciembre.

1 En la **Parte 3** de la práctica el **Process Generator** se convertirá en **Loader**, que en vez de generar procesos al azar, leerá programas de un fichero de texto que deberá cargarlos en memoria y gestionar las estructuras de datos necesarias.

2 El parámetro tiempo de vida desaparecerá en la **Parte 3** de la práctica.