Universidad de Alcalá

SIMULACIÓN DEL CONTROL DE PROCESOS EN UN SISTEMA OPERATIVO

Parte I – Planificador con Colas de Prioridad[1]

En esta práctica se pretende simular el funcionamiento de un planificador de procesos de un sistema operativo. En esta primera parte, simularemos el funcionamiento de un planificador con tres núcleos. El objetivo será calcular el tiempo medio que los procesos pasan en el sistema. Como resultado final se busca calcular el tiempo medio que los procesos se encuentran vivos dentro del sistema operativo.

El programa principal simulará el funcionamiento de un planificador de procesos, implementando[2] los núcleos, procesos, Pila (con sus operaciones habituales) y Cola (con sus operaciones habituales y el método insertar por prioridad).

Cada **proceso** que se inicio dentro del sistema operativo tendrá los siguientes datos:

- **Identificador del proceso** (PID).
- **Identificador del proceso padre (PPID).** Por defecto init con PID 1.
- **Inicio del proceso** (en minutos desde las 00:00).
- **Tiempo de vida** (en minutos). Los procesos deben indicar cuanto tiempo se encontrarán en ejecución.
- **Prioridad del proceso** (cuanto menor es el valor de prioridad, mayor prioridad tiene el proceso). Las prioridades de los procesos vendrán dadas por un rango número entre 0 y 9.
- **Núcleo asignado** para la ejecución cuando se le asigne tiempo de CPU.

El programa funcionará creando manualmente en el código un mínimo de 10 procesos, que se almacenarán en una pila ordenada por hora de inicio de cada proceso. Los procesos se atenderán por orden de prioridad en función de sus características, similar a una cola de prioridad.

Ejemplo de ejecución:

- **Inicio de un proceso**: El proceso será asignado a un núcleo libre o se añadirá a la cola de espera si todos los núcleos están ocupados.
- **Finalización de un proceso**: Cuando un proceso termina su ejecución, el núcleo quedará disponible para el proceso con mayor prioridad en la cola de espera.

---

[1] Una cola de prioridades es un tipo de dato abstracto similar a una cola en la que los elementos tienen adicionalmente, una prioridad asignada. En una cola de prioridades un elemento con mayor prioridad se sitúa antes, y por tanto será desencolado antes, que un elemento de menor prioridad. Si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de cola.

[2] La implementación se realizará utilizando Programación Orientada a Objetos y memoria dinámica.

Universidad de Alcalá

El programa funcionará mediante un menú con, al menos, las siguientes opciones:

1. **Crear la pila de procesos** que se iniciarán en el sistema. Los datos se crearán manualmente en el código.
2. **Mostrar** los proceoss que se iniciarán (**pila de procesos**).
3. **Borrar** la **pila de procesos** que se van a ejecutar en el sistema operativo.
4. **Mostrar** la **cola de espera** de procesos.
5. **Mostrar** los detalles de los procesos que se encuentran **en ejecución en cada uno de los nucleos**. (PID del proceso, PPID del proceso, y tiempo
6. Simular que han pasado **N minutos** (N leído de teclado). Se simulará el paso del tiempo, escribiendo en pantalla los datos de los eventos de llegada y/o salida de procesos que vayan realizándose (minuto, PID, PPID, id del núcleo que ejecuta el proceso, estado de los distintos núcleos en ese momento).
7. Simular el **funcionamiento y ejecución de todos los procesos** que se encuentran en el sistema operativo, es decir, hasta que no quede ningún proceso sin ejecutarse ni en espera. Se simulará el paso del tiempo, escribiendo en pantalla los datos de los eventos de inicio y/o finalización de procesos que vayan realizándose (minuto, PID, PPID, id del núcleo que ejecuta el proceso, estado de los distintos núcleos en ese momento). Al finalizar se mostrará el tiempo medio de estancia en el sistema operativo.

Universidad de Alcalá

Parte II – Listas

En esta segunda parte se implementará el TAD lista para gestionar los núcleos (cores) del sistema operativo, de forma que en cada momento se utilicen tantos cores como sea necesario. Simulando un entorno en la nube con capacidad de elasticidad y adaptabilidad.

Gestionándose de esta forma una lista de núcleos (cores) que se adaptará a la carga del sistema, añadiendo o eliminando núcleos según la necesidad.

Cada núcleo incluirá, además de su id y el proceso (PID) que está siendo ejecutado, una cola de procesos en espera. Cuando se produzca un evento de llegada, el proceso entrará directamente (en función de su prioridad y sin pasar por una cola previa) en el núcleo cuya cola sea más corta en ese momento (no teniendo por qué ser la cola que menos tarde en ejecutarse). Y será atendido cuando le corresponda en ese núcleo. Los procesos no pueden cambiarse de núcleo, y por tanto de cola, aunque haya núcleos vacíos.

Se recomienda a fin de guardar de manera coherente toda la información, ordenar la *lista* por el código identificativo de los núcleos.

Se pide implementar las siguientes operaciones:

- Añadir un núcleo cuando todos los núcleos tienen más de 2 procesos en espera.
- Eliminar núcleos vacíos cuando haya al menos dos núcleos libres. (la lista siempre debe tener un núcleo)
- Consultar en cualquier momento cuál es el núcleo más ocupado y el que tiene menos procesos.
- Consultar el número total de núcleos operativos.

El programa funcionará ahora mediante un **menú** con, al menos, las siguientes opciones:

1. **Crear la pila de procesos** que se ejecutaran en el sistema operativo, los datos se crearán manualmente en el código.
2. **Mostrar** los proceoss que se iniciarán en el sistema operativo (**pila de procesos**).
3. **Borrar** la **pila de procesos** que llegarán al sistema opeartivo.
4. Simular que han pasado **N minutos** (N leído de teclado). Se simulará el paso del tiempo, escribiendo en pantalla los datos de los eventos de inicio y/o fin de procesos que vayan realizándose (PID, PPID, id del núcleo que ejecuta el proceso, estado de los distintos núcleos en ese momento).
5. **Mostrar** los datos de la **lista de núcleos** (id del núcleo, proceso que esta siendo ejecutado, tiempo restante de ejecución, procesos que están a la espera de ser ejecutados)
6. **Consultar** en cualquier momento que **núcleo** tiene **menos procesos** y cuál es el **más ocupado**.
7. **Consultar** en cualquier momento el **número de núcleos** de atención **operativos**.
8. Simular el **funcionamiento del sistema operativo ejecutando todos los procesos**, es decir, hasta que no quede ingún proceso sin ser ejecutado y en espera. Se simulará el paso del tiempo, escribiendo en pantalla los datos de los proceos de inicio y/o fin de procesos que vyan realizandose (PID, PPID, núcleo en el

Universidad de Alcalá

que entra/sale, estado de los distintos núcleos en ese momento). Al finalizar, se mostará el tiempo medio de estancia de los procesos en el sistema operativo.

OBSERVACIONES:

- Ambas partes **deben funcionar por separado**, se recomienda implementar la primera y que funcione correctamente antes de comenzar la segunda.
- Aquí se está describiendo el funcionamiento básico del programa y las estructuras correspondientes, los alumnos deben tomar decisiones sobre los datos, su implementación, la interfaz del programa, etc, que deben ser justificadas en la documentación.
- Es decisión de cada grupo la forma en que se mostrará en pantalla el contenido de las estructuras y la información sobre el funcionamiento de la simulación, se valorará la claridad de la misma.
- Una vez entregada la práctica, se realizará una **defensa** en la que el programa debe **funcionar correctamente con los datos de prueba** que se facilitarán. La defensa, **individual y obligatoria para ser calificado**, constará de dos partes:
    - Defensa escrita (se realiza sin máquina) se puede traer, si se desea, el código en papel.
    - Defensa oral: para su realización es necesario traer el código entregado en el portátil o en un pen drive (en caso de no traer el portátil).