CC3002 – Proyecto 2

I. Modalidad de trabajo. Individual

II. Calendario

a) Asignación: jueves 25/02/2010 durante el período de clase.
b) Dudas: jueves 11/03/2010 durante el período de clase.
c) Entrega: miércoles 17/03/2010 antes de clase, a través de sakai.

III. Objetivos

a) Que el estudiante aplique los conceptos aprendidos en clase sobre:
  i) Algoritmos de calendarización de CPU
b) Crear las bases para los siguientes proyectos del curso.

IV. Ambiente de trabajo. El proyecto se implementará en base al Proyecto 1. Es importante que antes de que inicie este proyecto, el Proyecto 1 funcione bien, en especial:

a) Calendarización de procesos en ambiente multiprogramación.
b) I/O. Colas para acceder a archivos.

V. Instrucciones. El proyecto consiste en extender el S.O. para permitir que ejecute procesos utilizando diferentes algoritmos de calendarización.

a) Random. Esta instrucción devolverá un número aleatorio en el registro especificado.

random r1 r2 r3. Devuelve en *r3* un valor aleatorio entre *r1* y *r2*. Ejemplo: si *r1* = 1 y *r2* = 100, *r3* devuelve un valor aleatorio entre 1 y 100 (inclusive).

b) Prioridad. Cada proceso tendrá una prioridad asociada, la cual será utilizada en ciertos algoritmos de calendarización. Entre más grande sea el número, mayor será la prioridad del proceso. Las prioridades estarán definidas entre 0 – 9 (inclusive), de forma que la máxima prioridad de un proceso podrá ser 9 y la menor podrá ser 0.

La prioridad será asignada en el momento que el usuario ejecute un *user program* en la línea de comando. Si la prioridad no es especificada, el valor por defecto será 4. En el caso de *system programs*, la prioridad es omitida (ya que los *system programs* no son calendarizados). En el *fork* también es posible especificar la prioridad.

Fibonacci 10 (Se asume prioridad = 4)
Fibonacci 10 p1
Fibonacci 100 > fib.txt p5

c) System Calls. Las system calls que hacen pasar un proceso del estado running a waiting (luego a ready), tendrán un tiempo de realización. Es decir, ya no serán inmediatas. Aplica para las siguientes *system calls*:

i) OpenFile
ii) CloseFile
iii) ReadFile
iv) Sleep
v) Echo

Por ejemplo, en el *tick21*, el *proceso p* ejecuta una de las *system calls* anteriores. Al finalizar el *tick21*, el proceso pasa a estado *waiting*, donde espera 5 *ticks* antes de pasar al estado ready.

Tick21	Tick22	Tick23	Tick24	Tick25	Tick26	Tick27
running	waiting	waiting	waiting	waiting	waiting	ready

Nota: La *system call* (abrir archivo, lectura de archivo, etc…) se puede efectuar durante el *tick21* y no necesariamente hasta el *tick26*.

d) **Calendarización.** El S.O. podrá calendarizar procesos de acuerdo al modelo *Multilevel Feedback Queue* visto en clase. El algoritmo estará compuesto por varias colas, de forma que el CPU se asignará a procesos de la *n-ésima* cola, si y solo si, no existen procesos en estado *ready* en todas las colas anteriores (colas *1* a *n-1*) La configuración de cada una de las colas será cargada de un archivo XML al iniciar el S.O. El archivo xml tendá el nombre de "sys.xml". Cada cola estará definida según los siguientes parámetros.

- **i)** **id.** Número de cola.
- **ii)** **name.** Nombre de la cola.
- **iii)** **type.**
  - **(1)** FCFS
  - **(2)** Prioridad
  - **(3)** SJF (Shortest job first). Este parámetro debe ser calculado por el estudiante a su criterio. El estudiante debe definir una fórmula la cual puede tomar en cuenta variables como:
    - (a) Longitud del programa (líneas).
    - (b) % de líneas del programa que representan instrucciones y no *system calls*. Indicador de CPU bursts largos.
    - (c) Número de archivos abiertos. (entre más archivos abiertos tenga un proceso, mayor la probabilidad de que los CPU bursts sean cortos)
    - (d) Etc…
- **iv)** **quantum.** Número máximo de ticks que un proceso puede tener el CPU. 0 si no hay máximo.
- **v)** **preemptive.** 1 o 0. Indica si procesos nuevos (o procesos que regresan a estado *ready*) pueden forzar al proceso que se esté ejecutando a cederle el CPU (En el caso que tenga mayor prioridad o valor de SJF). El parámetro es ignorado en el caso de FCFS.
- **vi)** **queueOnQuantumEnd.** Nueva cola (ID) para un proceso que ha terminado su quantum de CPU.
- **vii)** **queueOnIOEnd.** Nueva cola (ID) para un proceso que pasa del estado *waiting* a *ready*.

e) **Estadísticas.** El S.O. debe mostrar ciertas estadísticas con respecto a la calendarización de los procesos. El estudiante puede generar archivos de Excel, mostrar gráficas de pie, etc…

- **i)** Estadisticas del S.O. en base a procesos terminados.
  - **(1)** Utilización de CPU
  - **(2)** Throughput (Procesos terminados / Número de ticks)
  - **(3)** Turnaround time promedio
  - **(4)** Waiting time promedio
  - **(5)** Response time promedio
  - **(6)** Execution time promedio
- **ii)** Estadísticas por proceso.
  - **(1)** Turnaround time
  - **(2)** Waiting time
  - **(3)** Response time
  - **(4)** Execution time

f) **Semáforos.** El S.O. proveerá semáforos a los procesos por medio de *system calls*:

- **i)** **wait** reg Si ningún proceso está esperando bajo el semáforo con número especificado (reg), el proceso continúa la ejecución. De lo contrario, se encola hasta que todos los procesos que invocaron *wait* anteriormente, hayan invocado *signal*.
- **ii)** **signal** reg**.** El proceso señala que ha dejado de utilizar el semáforo con número especificado (reg). Debe "despertar" al siguiente proceso que haya ejecutado *wait*.

g) **Logs.** Asegúrese de que los *logs* del S.O. sean actualizados según las nuevas características del mismo.

- **i)** **io.log.**
- **ii)** **ps.log.**

VI. Presentación. Todos los estudiantes deben llegar preparaos para presentar el primer día de presentación del proyecto. El orden de presentación se rifará al inicio de la clase. En este momento, todos los proyectos deben estar subidos en sakai. La presentación del proyecto incluye dos fases:

- **a)** Presentación de los algoritmos / estructuras más importantes. Si el estudiante considera necesario, puede elaborar diagramas que lo ayuden a explicar mejor. (Aproximadamente 5 minutos)
  - **i)** Algoritmo del calendarizador de procesos (Cambios de proceso, cambios de estado, ejecución de proceso).
  - **ii)** Definición de fórmula de SJF.
  - **iii)** Consideraciones de puntos extra.
- **b)** Pruebas del S.O. (Aproximadamente 5 minutos)

VII. Evaluación. Los puntos del proyecto están distribuidos de la siguiente manera:

Descripción	Porcentaje
Random	5 %
Prioridad	5 %
System calls con wait	10 %
Calendarización	30 %
Estadísticas	15 %
Semáforos	10%
Logs	5 %
Presentación del proyecto	20 %
Total	100.00%

¡PUNTOS EXTRA! Cualquier aspecto adicional a lo pedido en el proyecto, que sea de **utilidad** en el programa y tenga **mayor dificultad** de programación de lo pedido, se tomará en consideración para puntos extra, teniendo como **MÁXIMO de 10 PUNTOS EXTRA**.

VIII. Recomendaciones

- **a)** **Dificultad.** El proyecto no es fácil y va a tomar tiempo desarrollarlo. No lo deje a última hora.
- **b)** **Modularidad.** Trabaje lo más modular y ordenado posible, ya que el resto de proyectos se implementarán en base a este proyecto.

IX. Material de apoyo.

- **a)** SAKAI http://sakai.uvg.edu.gt Curso: CC3002SistOperativos
- **b)** Silberschatz, Galvin, Gagne. Operating System Concepts. Seventh Edition. Wiley.