CC3002 – Proyecto 3

I. Modalidad de trabajo. Individual

II. Calendario
- **a) Asignación**: jueves 8/04/2010 durante el período de clase.
- **b) Dudas**: todos los días de clase.
- **c) Entrega**: miércoles 5/05/2010 antes de clase, a través de sakai.

III. Objetivos
- **a)** Que el estudiante aplique los conceptos aprendidos en clase sobre:
  - **i)** Threads
  - **ii)** Memoria
  - **iii)** Swapping y paginación
- **b)** Crear las bases para los siguientes proyectos del curso.

IV. Ambiente de trabajo. El proyecto se implementará en base al Proyecto 1 y el Proyecto 2. Es importante que antes de que inicie este proyecto, ambos funcionen bien, en especial:

- **a)** Calendarización de procesos utilizando round robin (FCFS con q=5). Se utilizará este algoritmo de calendarización como estándar.

V. Instrucciones. El proyecto consiste en extender el S.O. para permitir a los procesos acceder a memoria de datos.

- **a) Registros.** El número de registros ahora será de 16 (r0 a r15).

- **b) Memoria de datos.** El S.O. contará con un número de frames en ram y un número de frames en forma de swap. El número de frames, así como el tamaño de frame, estarán definidos en el archivo *sys.xml* de la siguiente forma:

```
<sys>
        <memory ram="8" swap="16" page="16" />
        <scheduling startQueue="1">
        …
        </scheduling>
</sys>
```

La memoria se utilizará únicamente para almacenar datos, no almacenará código de los procesos. La memoria estará compuesta por un arreglo de enteros, cada uno identificado con una dirección de memoria. Al crear un proceso (desde línea de comando, fork o el comando thread) y no se cuente con la cantidad de frames especificadas, el S.O. no creará el proceso y mostrará un mensaje de error.

El número de páginas (tamaño de memoria) (memoria lógica) de cada proceso será especificado al crearlo desde línea de comando o fork. Si el número de páginas no es especificado, este iniciará con dos páginas.

Fibonacci 10 (Se asume número de páginas = 2)
Fibonacci 10 m1 (número de páginas = 1)
Fibonacci 10 p1 m1 (número de páginas = 1)
Fibonacci 100 > fib.txt m5 (número de páginas = 5)
Fibonacci 100 > fib.txt p5 m5 (número de páginas = 5)

Los datos en memoria únicamente podrán ser accedidos a través de dos instrucciones (No son system calls):

ReadMem *reg1 reg2*. Almacena el valor de la dirección lógica de memoria dada por el valor de *reg1* en *reg2*.
WriteMem *reg1 reg2*. Escribe el valor de *reg2* en la dirección lógica de memoria dada por el valor de *reg1*.

Nota: Si la dirección lógica es inválida, debe terminar el proceso y devolver error (Debe estar comprendida entre 0 y max-1).

c) **Threads.** Funciona de la misma manera que funciona la system call fork. Crea un proceso hijo con el programa especificado, prioridad, método de despliegue (sincrónico: consola, ó asincrónico: archivo). La única diferencia es que el proceso hijo y el proceso padre compartirán las mismas páginas en memoria.

i) **Thread** *string reg1*.

d) **Swapping.** Durante el context switch, el S.O. debe garantizar que las páginas del proceso a quien se le cederá el CPU, estén mapeadas a un frame en memoria ram, no en memoria swap (virtual). Para lograr este objetivo, el S.O. deberá contar con las siguientes estructuras:

i) Memoria principal. Arreglo de frames. Para el proceso en control del CPU, cada una de sus páginas debe estar mapeada a una frame en memoria principal.
ii) Memoria virtual. Arreglo de frames, almacenadas temporalmente.
iii) Tabla de paginación. Utilizada para la traducción de una dirección de memoria lógica a una dirección de memoria física.
iv) Tabla de frames: Utilizada para la administración de frames asignadas y disponibles.

El S.O. para elegir la siguiente frame en memoria principal a realizar swap out (debido a que un nuevo proceso se le fue asignado el CPU y sus páginas no están mapeadas a frames en memoria principal) deberá tomar en cuenta:

i) Frames no utilizadas.
ii) Frames asignadas anteriormente

b) System call **top,** el cual desplegará los contenidos de cada frame en memoria principal y en memoria swap (virtual). Adicionalmente, debe mostrar el % de asignación tanto para memoria principal, como para memoria swap (virtual).

c) **Logs.** Asegúrese de que los *logs* del S.O. sean actualizados según las nuevas características del mismo. Adicionalmente, deberá incluir el log *mem.log*.

i) **io.log.**
ii) **ps.log.**
iii) **mem.log.** Guarda todos los cambios en la tabla de frames/paginación.

II. **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)
b) Pruebas del S.O. (Aproximadamente 5 minutos)

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

Descripción	Porcentaje
Memoria de datos	30 %
Threads	10 %
Swapping	30 %
Top	10 %
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**

IV. 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.

V. Material de apoyo.

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