CC3002 – Proyecto 1

I. Modalidad de trabajo. Individual

II. Calendario
- **a) Asignación**: jueves 28/01/2010 durante el período de clase.
- **b) Dudas**: jueves 11/02/2010 durante el período de clase.
- **c) Entrega**: miércoles 17/02/2010 antes de clase, a través de sakai.

III. Objetivos
- **a)** Que el estudiante aplique los conceptos aprendidos en clase sobre:
  - **i)** Interfaz de usuario de un S.O.
  - **ii)** System calls
  - **iii)** Ejecución de procesos
  - **iv)** Ambiente multiprogramación para la ejecución de procesos
- **b)** Crear las bases para los siguientes proyectos del curso.

IV. Ambiente de trabajo. El proyecto puede implementarse en cualquier Sistema Operativo y en cualquier lenguaje de programación de alto nivel.

V. Instrucciones. El proyecto consiste en elaborar una aplicación que simule el comportamiento de un sistema operativo. Deberá tener los siguientes elementos:

a) Línea de comando. La interfaz para el usuario será una línea de comando. A través de ésta, el usuario podrá ejecutar system programs o user programs. Ambos pueden recibir parámetros, los cuales son separados por espacios. La línea de comando tendrá un directorio donde se estará trabajando. Por el momento, el S.O. únicamente contendrá un directorio raíz, el cual deberá ser mapeado a un directorio del S.O. real. De forma que los archivos que el estudiante coloque en dicho directorio en el S.O. real deberán ser visibles dentro del S.O. simulado. Por el momento no se manejarán carpetas.

El usuario puede especificar el modo de output del programa:

- **i) Output a línea de comando (Síncrono).** Ejemplo: kill 451. La línea de comando muestra el output del programa y se bloquea (No permite el ingreso de nuevos comandos).
- **ii) Output a archivo (Asíncrono).** Ejemplo: kill 451 > kill.log. Invoca al programa kill con el parámetro 451, el output se escribe (si el archivo existe, debe escribir al final) en el archivo kill.log. La línea de comando permite el ingreso de nuevos comandos.

b) System programs. Estos programas deben ser implementados por el estudiante y podrán ser invocados por el usuario a través de la línea de comando. El estudiante debe implementar al menos los siguientes system programs:

- **i) ls.** Muestra los archivos dentro de la carpeta que se está trabajando. No recibe parámetros.
- **ii) kill.** Termina el proceso con pid indicado.
- **iii) ps.** Muestra todos los procesos activos dentro del S.O. Además, debe mostrar los archivos que cada proceso tiene en uso.

c) User programs. El S.O. podrá ejecutar programas creados por el usuario. Al ejecutar un programa, se crea un proceso.

- **i) Parámetros.** El máximo número de parámetros será de 8 ya que cada registro es inicializado con el parámetro. R0 – param1, R1 – param2, etc…

- **ii) Instrucciones y registros**

  - **(1) 8 Registros: R0, R1…R7.** Los registros podrán almacenar valores enteros de 32 bits con signo.

  - **(2) Instrucciones**

    (a) **Set** *reg valor* o set reg1 reg2. Ejemplo: set r1 0. Establece el valor especificado en el registro.

(b) **Add** *reg1 reg2 reg3*. Equivalente a reg1 = reg2 + reg3. Ejemplo: add r1 r2 r3.
(c) **Sub** *reg1 reg2 reg3*. Equivalente a reg1 = reg2 - reg3.
(d) **Mul** *reg1 reg2 reg3*. Equivalente a reg1 = reg2 * reg3.
(e) **Div** *reg1 reg2 reg3 reg4*. Equivalente a la división entera de reg1 = reg2 / reg3. reg4 contiene el residuo. En el caso que se efectúe una división en 0, el programa debe terminar con resultado -2.
(f) **Jmp** *reg lab*. Si reg es verdadero (mayor que 0), salta a lab. No es necesario poner delimitar la etiqueta con comillas simples. En el caso que la etiqueta no exista, el programa debe terminar con resultado -1.

(3) Etiquetas. Las etiquetas se utilizan para realizar saltos. Están compuestas por caracteres alfanuméricos y deben ir delimitados por comillas simples (‘). Las líneas que contengan una etiqueta, no podrán contener una instrucción o system call. Ejemplo de una línea: *‘loop’*. En este caso, al saltar a la etiqueta loop, la siguiente instrucción a ejecutar será la instrucción después de la línea de la etiqueta.

(4) Comentarios. Los comentarios ocupan una línea completa, es decir no puede existir un comentario donde exista una instrucción. Los comentarios deben iniciar con punto y coma (;).

d) Calendarización de Procesos. El S.O. utilizará multiprogramación para la ejecución de los procesos. Es decir, el procesador cambia de proceso únicamente si se realiza alguna operación de i/o o si el proceso termina. El siguiente proceso a ser ejecutado es el siguiente proceso dentro de la cola (Procesos con estado *ready*). Cada proceso debe tener uno de los siguientes estados:

i) New (stateId: 1). Cuando el proceso está siendo creado o está esperando que se le asigne memoria.
ii) Ready (stateId: 2). Cuando el proceso está listo para ejecutarse.
iii) Running (stateId: 3). Cuando el proceso está asignado al procesador.
iv) Waiting (stateId: 4). Cuando el proceso está esperando alguna solicitud de I/O.
v) Terminated (stateId: 5). Cuando un proceso hijo ha terminado, pero su proceso padre todavía está activo.

e) System Calls. El S.O. proveerá varias funciones que permitirán a los desarrolladores de user programs trabajar en conjunto con el S.O. En algunos casos, se envían registros como parámetros para obtener resultados.

i) Echo “string” o **Echo** reg**.** Despliega la cadena o el valor ascii del registro. Dependiendo de la forma en que el programa haya sido invocado, el despliegue se efectuará en la línea de comandos o en un archivo.
ii) OpenFile “nombre del archivo” reg. Abre el archivo con el nombre especificado (Espera si el archivo está en uso). El registro devuelve el valor del handle (Identificador de archivo), 0 si no fue posible abrir el archivo. El proceso debe pasar al estado waiting.
iii) CloseFile reg. Cierra el archivo especificado. El proceso debe pasar al estado *waiting*.
iv) ReadLine reg1 reg2. Lee una línea del archivo y lo imprime. Es decir, leer desde la posición actual hasta (inclusive) un carácter de nueva línea. Reg1 especifica el handle y reg2 devuelve el número de caracteres impresos. El proceso debe pasar al estado *waiting*.
v) Sleep. El proceso cede el procesador al siguiente proceso. El proceso debe pasar al estado *waiting*.
vi) End valor o **End** reg. Termina el proceso con el resultado especificado. Se utilizarán valores positivos para resultados exitosos y negativos para errores. 0 se utilizará para procesos no terminados en el caso de GetState.
vii) Fork “string” reg. Crea un proceso, la cadena especificada contiene el nombre del comando (system program o user program) y los parámetros. El proceso hereda el mecanismo de output (línea de comando o archivo). En el registro especificado se devuelve el *pid* del proceso creado.
viii) GetState *valor reg2 reg1* o **GetState** reg3 *reg2 reg1*. Obtiene el estado del proceso especificado (stateId) en reg2. Retorna el resultado (Si el proceso no ha terminado, devuelve 0) en reg1. Únicamente será posible obtener el estado de un proceso hijo.

f) I/O. En este proyecto, los archivos únicamente pueden ser abiertos por un proceso a la vez y únicamente pueden ser leídos. Por cada archivo que esté en uso, es necesario almacenar el puntero de archivo (Puntero a la siguiente posición a leer) .Adicionalmente, debe existir una cola con los procesos que deseen acceder a cada archivo abierto.

g) Logs. El S.O. manejará los siguientes logs:

i) io.log. Contendrá todas las peticiones de abrir (cola de procesos para acceder archivos) y cerrar archivos.
ii) ps.log. Contendrá las acciones relacionadas a los procesos:

(1) Creación y terminación de procesos.
(2) Cambio de estado.

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) Estructura del PCB.
  ii) Colas del calendarizador de procesos.
  iii) Algoritmo del calendarizador de procesos (Cambios de proceso, cambios de estado, ejecución de proceso).
  iv) Consideraciones de ptos extra.
b) Pruebas del S.O. (Aproximadamente 5 minutos)

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

Descripción	Porcentaje
Línea de comando	15 %
System programs	10 %
User programs	15 %
Calendarización de procesos	15 %
System Calls	15 %
I/O	5 %
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.