Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Sistemas Operativos 1 Sección B
Ing. Francisco Guevara
Aux. Fernando Samayoa

Proyecto Final

Simulador de procesos

Objetivos

- Implementar algoritmos de planificación y prevención de interbloqueos
- Simular el funcionamiento del administrador de recursos del sistema
- Desarrollar los algoritmos de solución de interbloqueo.

Descripción

Simulador

Simulación de administrador de recursos

Lenguaje de administración de recursos

Se define un lenguaje como texto de entrada para simular en el tiempo la solicitud de recursos al sistema por distintos procesos.

```
Simulación {

        //Declaración de Recursos
        //Ejecución de tiempos
}
```

Recurso

Es una entidad virtual genérica que puede ser asignada a un proceso. Los recursos deben ser declarados en la parte inicial del código para luego ser utilizados.

Sintaxis	Ejemplo
Recurso R[No. Recurso]	Recurso R1; Recurso R2; Recurso R3;

Tiempo

Se refiere al tiempo de CPU o tiempo de ejecución, se utiliza como métrica para la duración de la ejecución de un proceso. Los tiempos no necesitan declaración únicamente se ejecutan secuencialmente

Sintaxis	Ejemplo

Tiempo([No. de Tiempo]) { }	Tiempo(1){ //Acciones en procesos }
	Tiempo(2){ // Acciones en procesos }

Proceso

Es una unidad de una actividad que ejecuta una secuencia de instrucción utilizando recursos. Los procesos no se declaran únicamente se invocan para ejecutar una acción. No existen invocaciones sin acciones.

Acciones

Sintaxis	Ejemplo
Proceso([No. de Proceso) { }	Proceso(5){ //Asignacion y Liberacion de Recursos }

	Proceso(4){ //Asignacion y Liberacion de Recursos }

Sintaxis	Ejemplo
Asignar ([No. Recurso])	Asignar(R2,R6,R7);
Liberar ([No. Recurso])	Liberar(R1,R2,R3);
Terminar()	Terminar()

Ejemplo:

```
Proceso (5) {
        Asignar (R2,R6,R7);
        Liberar (R1,R2,R3);
}
```

Ejemplo General:

```
Simulacion
{
    Recurso R1;
    Recurso R2;
    Recurso R3;

    Tiempo(1)
    {
        Proceso(1)
        {
            Asignar (R1);
        }
        Proceso(2)
        {
            Asignar(R2);
        }
        Proceso(3)
        {
            Asignar(R2,R3,R4);
        }
    }
    Tiempo(2)
    {
        Proceso(4)
        {
            Asignar(R1,R6);
        }

    }
    Tiempo(3)
    {
        Proceso(1)
        {
            Asignar(R6);
        }

    }
    Tiempo(4)
    {
        Proceso(3)
        {
            Liberar(R3);
        }

    }
    Tiempo(5)
    {
        Proceso(3)
        {
            Terminar();
        }

    }
```

Manejo de interbloqueos

El propósito de la definición de procesos y recursos es aplicar las diferentes estrategias de solución de interbloqueos en un sistema operativo:
Las condiciones para el interbloqueo son las siguientes: **exclusión mutua, retención y espera, no apropiación y espera circular.**

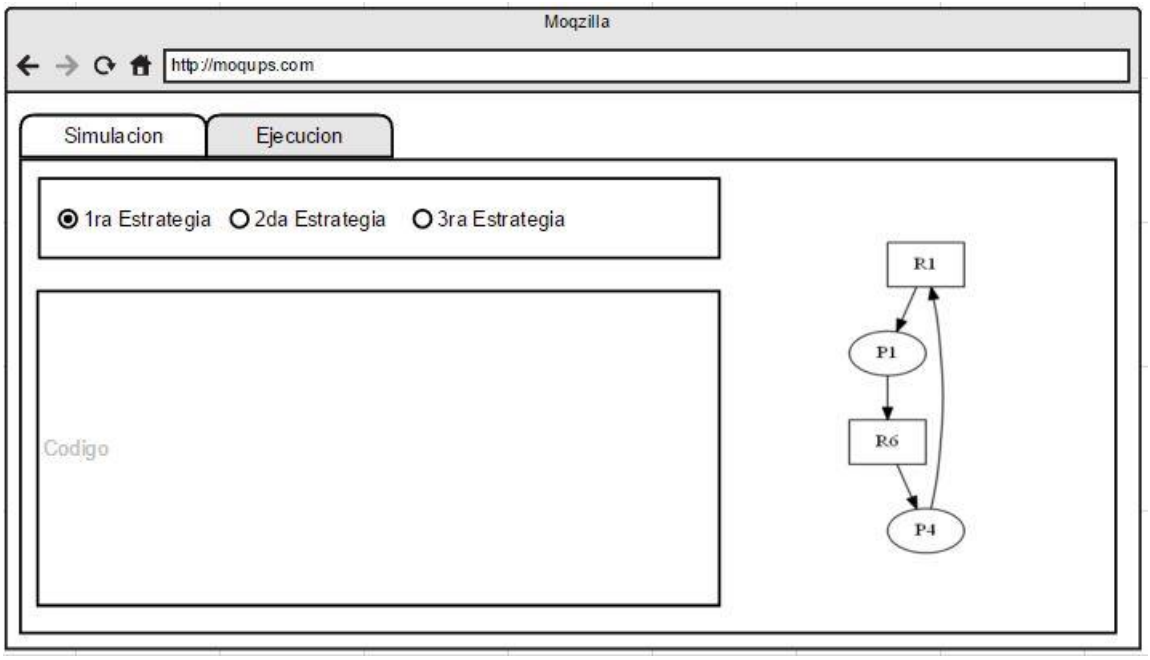
- **Detección y solución de interbloqueos**
    - Solución de interbloqueo por eliminación
        - Por consumo de recursos (el que más recursos consume)
        - Por el más viejo (el que entro primero de los que conforman el interbloqueo)
        - Por el ultimo que entro.

    NOTA: para la detección se debe alertar que se encontró un interbloqueo y mostrar el grafo de asignación recurso-proceso, destacando en donde se encuentra el circulo de espera.

- **Prevención**
    - *Como son técnicas de prevención, cambian la lógica del programa, pues estas condicionan al sistema operativo y a sus procesos para que los recursos se pidan o se manejen de una manera especial.*
    - Supresión de retención y espera: se debe asignar todos los recursos desde el inicio de la ejecución del proceso. **1era estrategia de havender**
    - Negación de la apropiación: si un proceso está en ejecución y necesita otro recurso que no puede obtener, libera todos sus recursos. **2da estrategia de havender**
    - Negación de espera circular: la asignación de recursos solo se puede dar en un orden (ascendente o descendente), con el fin de evitar el interbloqueo. **3ra estrategia de havender.**
    - Para la 3ra estrategia, se debe poder cambiar el orden, ascendente o descente de asignación.

    NOTA para todas las estrategias de prevención, se debe mostrar en tiempo real el grafo de recurso-proceso.

Vista



Simulación de planificador de procesos

Lenguaje de Simulación de Procesos

Ejecucion {

//Declaración de Procesos

}

Los procesos no poseen recursos únicamente interesa la implementación de los algoritmos de planificación y el grafico de ejecución de los mismos, por lo tanto basta con la declaración de los procesos.

Proceso

Es una unidad de una actividad que ejecuta un segmento de tiempo de CPU específico, contiene una prioridad de ejecución y un tiempo de vida.

Declaración

- No. de proceso: Identificador numérico único del proceso.
- Tiempo de llegada: El tiempo de CPU cuando el proceso empieza a funcionar.
- Tiempo de duración: El tiempo que el proceso está “vivo”.
- Prioridad: Prioridad de ejecución.

Sintaxis	New Proceso([No. de Proceso],[Tiempo de llegada],[Tiempo de duración],[Prioridad]);
Ejemplo	New Proceso(4,1,3,2); New Proceso(5,2,4,5); New Proceso(5,2,3,5);

Ejemplo general:

```
Ejecucion {

    Proceso(3,3,1,4);
    Proceso(5,5,5,5);
    Proceso(4,6,5,6);
    Proceso(6,9,1,5);
    Proceso(1,10,1,3);
    Proceso(2,11,5,5);

}
```

Algoritmos de planificación

*Para poder determinar en qué orden se ejecutarán los procesos si estos se encuentran en estado **listo**, simulara el **dispatcher** y el algoritmo que utiliza para poder realizar sus acciones.*

- **FCFS** primer proceso de llegada es el que es atendido, si un proceso llega mientras otro es ejecutado, debe esperar para poder ser ejecutado. (no expulsiva)
- **SCF** (shortest job first) selecciona el proceso de la cola de espera, cuyo tiempo de ejecución es el menor, en caso de haber empate, se selecciona por FCFS. (no expulsiva)
- **Prioridad** se ejecuta el proceso, cuya prioridad sea mayor en los procesos de la lista de espera. (no expulsiva)
- **RR** (round robín) Primero se define el número de **Quantum** el cual es el tiempo en que se le asignara el proceso al procesador, los procesos se colocan en una lista circular, si el tiempo de ejecución es mayor que el número de **Quantum** el proceso se debe reingresar a la lista, pero se debe descontar el tiempo de ejecución que ya paso, solo terminara de ejecutarse si el t de ejecución es 0 (expulsiva).

Vista

Reportes

Se refiere a la sección de reportes sobre el sistema operativo en uso, esto para conocer el sistema operativo Gnu/Liux, donde se deberán crear módulos de kernel para obtener información de la Ram y los procesos, de la misma manera que las instrucciones de la línea de comandos. No está permitido leer archivos que sean generados por módulos ya existentes, deben crear sus propios archivos de salida para cada modulo.

Módulo de Kernel

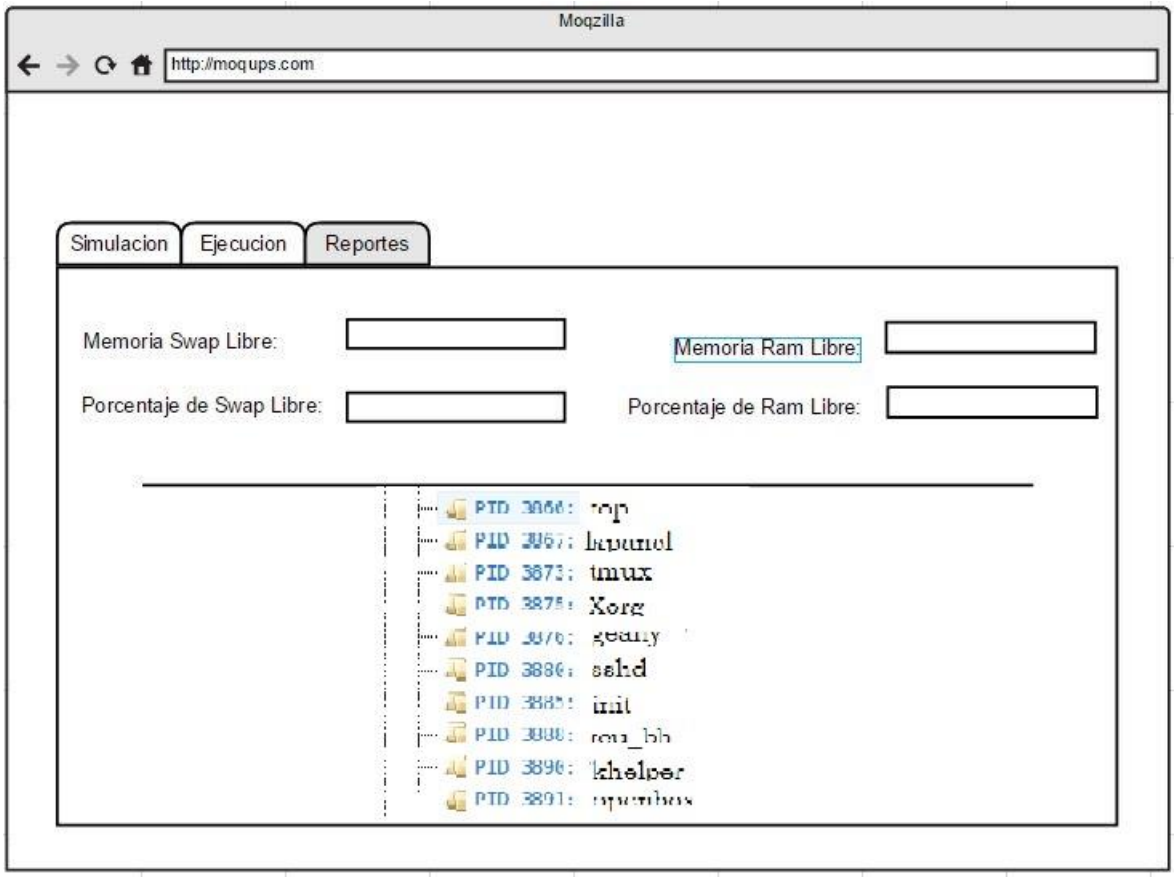
Módulo de Procesos

- Nombre del proceso
- PID
- Estado

Información de memoria

- Memoria Swap Libre
- Memoria Ram Libre
- Porcentaje memoria Swap libre
- Porcentaje memoria Ram Libre

NOTA: Deben utilizar los módulos utilizados en la primera fase del proyecto. Es permitido hacer modificaciones a la salida del módulo (contenido del archivo) a su conveniencia.



Observaciones

- Se debe utilizar un sistema GNU/Linux.
- Lenguaje de programación: libre.
- Para tener derecho a calificación, se deben cumplir con las restricciones.
- La información de los procesos, debe ser obtenida por el archivo creado por el modulo del estudiante, **no se permitirá obtener esta información de alguna otra forma**.
- Las gráficas deben generarse y mostrarse en tiempo real, de no ser así, el alumno tendrá una nota de 0 en ese apartado.
- Se debe respetar la sintaxis de los lenguajes definidos.
- No se calificará el parser para el lenguaje de entrada, pues no es tema del curso.
- NO HABRA PRORROGA.
- Debe enviarse el link al repositorio de github con el código fuente y **manuales (usuario y técnico)** al correo del auxiliar.
- Copias totales o parciales, tendrán una nota de 0 puntos y será reportado a escuela de sistemas.

Restricciones

1. Algoritmos de interbloqueo
2. Algoritmos de planificación de procesos con su respectiva simulación.
3. Graficas en las simulaciones (necesario para poder calificar).
4. **La entrega del proyecto es OBLIGATORIA para poder aprobar el laboratorio de Sistemas Operativos 1.**

Fecha de entrega:

Sábado 5 de noviembre antes de las 12 de medio día. (Se calificara ese mismo dia)
Asunto: [SO1]Proyecto_#carnet
Correo: fredysamayoa5@gmail.com