Dibujo en blanco y negro

Descripción generada automáticamente con confianza mediaUNIVERSIDAD NACIONAL AUTÓNOMA DE Una caricatura de una persona

Descripción generada automáticamente con confianza mediaMÉXICO

Facultad de Ingeniería

Práctica 4:

Simulador de procesos y memoria – Parte 1

Profesor: M.I. EDGAR TISTA GARCÍA

Alumnos: Calderón Guevara César Yair

Maceda Patricio Fernando

Grupo de teoría: 02

Fecha de realización: 13/10/2021

Semestre: 2022-1

Objetivos

Implementar los conceptos del curso relacionados con procesos realizando una implementación que permita simular la ejecución de procesos en un sistema operativo.

Que el alumno pueda retomar sus habilidades de programación aplicándolo a un concepto fundamental del curso.

Introducción

Utilizar una computadora lleva un sin número de actividades y desarrollo que muchos usuarios no ven a simple vista, es cuando se profundiza que podemos conocer varios aspectos de la computadora como su composición y más complicadamente, la forma en la que trabajan para abrir aplicaciones, permitirnos conectarnos a internet o simplemente darnos un escritorio donde interactuar.

Recordando que el sistema operativo (SO) es el responsable de establecer un puente entre el hardware (con su respectivo software) y el usuario, es importante conocer cómo es que se permite esta interacción, para no rondar con explicaciones largas, esto es realizado por medio de procesos. Los procesos son, en resumidas cuentas, programas en ejecución y cada uno de estos procesos indican operaciones que, al realizarse, producen una acción o resultado que es interpretado y ejecutado por la máquina en cuestión.

La velocidad a la cual la computadora lee y ejecuta estos procesos actualmente es tan veloz (milisegundos en muchos casos) que es muy complicado para nosotros poder seguir en tiempo real el ciclo de vida de un proceso (identificado de acuerdo con la actividad que esté realizando). Por ello, en esta práctica construiremos un algoritmo que nos permita construir y representar con los elementos aprendidos en teoría, un simulador donde podamos observar con detenimiento el flujo de un proceso.

Análisis y Desarrollo del Programa

Para la realización del programa presentado se tomaron en consideración los siguientes conceptos y solicitudes para el desarrollo de la práctica.

* Un procesador solo puede ejecutar un proceso en un instante de tiempo, pero puede haber muchos procesos que pueden estar preparados y esperando a ser ejecutados.
* El usuario tomará el rol del SO, es decir, podrá interactuar con el programa creando, ejecutando y realizando otras operaciones con los procesos como listarlos o borrarlos.
* Tendremos una “memoria disponible” que nos determinará un límite en la cantidad de procesos que podremos crear (no podemos crear procesos infinitos).
* La interacción del usuario como SO se hará por medio de un menú que contenga las siguientes acciones:
  + - Crear proceso nuevo.
    - Ver estado actual del sistema.
    - Imprimir cola de procesos.
    - Ver proceso actual
    - Ejecutar proceso actual.
    - Pasar al proceso siguiente
    - Matar al proceso actual
    - Salir del programa

Así mismo, se tomaron en cuenta las especificaciones que cada una de estas acciones debía realizar, tales como liberar memoria, mostrar el nombre del proceso, mostrar instrucciones asignadas o las direcciones de memoria ocupadas por dicho proceso entre otras. También se contempló la información que componía cada proceso como su PID, el nombre del proceso y otras características como el número de líneas y memoria abarcada.

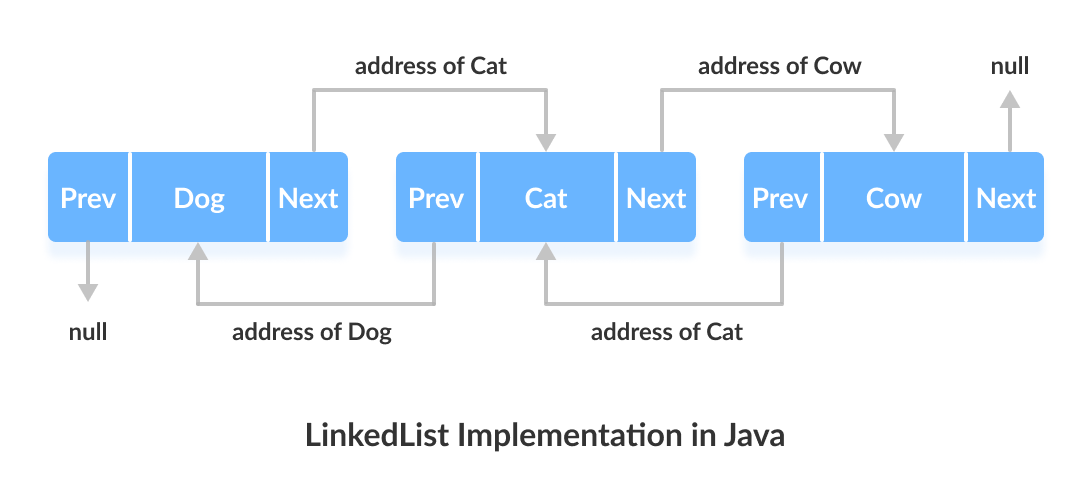
Tomando en cuenta lo anterior se llegó al acuerdo de realizar el programa en el lenguaje de programación Java para realizar este simulador, ya que es un lenguaje en el cual ambos participantes nos sentimos más familiarizados y así mismo, encontramos que podría ser implementado con mayor facilidad haciendo uso de las estructuras y el paradigma orientado a objetos (POO).

Después de una reunión en la plataforma Zoom, destacamos los puntos más importantes para la realización del programa, los cuales fueron los siguientes:

Texto

Descripción generada automáticamente

Donde el punto más importante fue el acuerdo de utilizar listas ligadas (Linked Lists), las cuales son una estructura de datos disponible en Java



Obtenida de: [Java LinkedList (instintoprogramador.com.mx)](https://www.instintoprogramador.com.mx/2019/12/java-linkedlist.html)

Con esta lista ligada dentro de una clase denominada *Memoria*, podremos realizar la secuencia entre los procesos creados y tenerlos en “orden” para poder interactuar con ellos con las acciones previamente identificadas. Sin embargo, recordemos que cada proceso tiene dentro de si un conjunto de información que debe ser almacenada por cada proceso generado como su PID, su nombre y su memoria ocupada, siendo esta la razón por la cual procederemos a utilizar la POO creando listas ligadas que almacenarán los procesos creados y con ellos, la información necesaria para interactuar con ellos.

Generaremos un total de 3 listas ligadas, la primera denominada *colaProcesos* será la responsable de almacenar todos los procesos creados y pendientes de ejecución, la segunda, denominada *finalizados*, guardará los procesos que hayan terminado su ejecución, finalmente, la tercera será denominada *eliminados* y contendrá todos aquellos procesos de los que se haya querido finalizar abruptamente su ejecución sin que hayan tenido oportunidad de finalizar o ejecutarse. Finalmente, recordaremos que la lista *colaProcesos* tendrá una memoria que fijará un límite en la cantidad de procesos que podremos crear, si esta memoria se acaba o se ve superada por otro proceso, no podrá crearse y tendremos que eliminar o finalizar un proceso ya encolado para liberar su memoria y agregar un nuevo proceso.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para los procesos se utilizará una clase *Proceso*, el cual tendrá como atributos el nombre del proceso, un *id*, sus instrucciones (un valor aleatorio), el espacio que ocupa (nuevamente, aleatorio para un conjunto definido) y las instrucciones ejecutadas. Gracias a la POO podremos realizar varias instancias de esta clase y cada uno podrá ser diferente en los atributos antes mencionados. Así mismo, cada proceso ocupará un porcentaje de memoria que se verá reflejado en la lista *colaProcesos* y en base a su tamaño se definirá si puede ser almacenado o no.

Texto

Descripción generada automáticamente

Después, en el código *main* se encontrarán las funciones que le permitirán al usuario programador interactuar como si fuera el SO, dándole la capacidad de generar procesos y gestionarlos con los puntos previamente mencionados. Para llevar a cabo lo anterior, se hizo uso de un menú que indicará la instrucción a realizar, lo único que el usuario debe hacer para seleccionar una instrucción es escribir el número correspondiente a la instrucción y pulsar enter. Mostraremos la sección del menú que nos dará la información sobre las acciones realizables:

Tabla

Descripción generada automáticamente con confianza media

Cada acción realizable será ejecutada por funciones que se encargarán de realizar la gestión entre las tres listas ligadas que representan la memoria, así como su contenido que serán los procesos y la información creada. Mostrando algunas de las acciones que realizarán invocaciones en el código:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Finalmente, estas funciones accederán a los objetos creados por las clases previamente definidas e interactuarán accediendo a sus atributos, modificando sus valores, mostrándolos o almacenando y eliminando nuevas instancias según la opción seleccionada. De manera general las funciones destacadas creadas son:

* Función menu(). Despliega el menú en terminal para que el usuario pueda escoger una acción, este menú se desplegará hasta que se seleccione la octava opción que permite salir del programa.
* Función crearProceso(). Permite crear un proceso instanciando un objeto de la clase Proceso. El nombre del proceso será asignado por el usuario en tiempo de ejecución y será asignado en la memoria *colaProcesos* si es que tiene espacio disponible.
* Función estadoActual(). Regresa información relevante del sistema como el contenido de las 3 colas, la memoria disponible y la asignación de localidades de memoria por cada proceso.
* Función mostrarProcesos(). Esta función permite enlistar en un formato de tabla los procesos creados y almacenados en la memoria *colaProcesos.* Los muestra en orden y el primer elemento mostrado es el proceso *activo*, es decir, aquel que será ejecutado.
* Función procesoActual(). Muestra el proceso activo, es decir, el primero en la memoria de *colaProcesos*.
* Función ejecutarProceso(). Realiza una interacción con «las instrucciones» de un proceso, simulando la ejecución de unas cuantas líneas de este y si no ha terminado de ejecutar el total de líneas, manda al proceso al final de la cola de procesos listos para ejecución, caso contrario, da el proceso por terminado y lo registrará en la lista de *finalizados.*
* Función siguienteProceso(). Manda el proceso activo al final de la lista de procesos listos para ejecutarse (*colaProcesos)* sin ejecutar ninguna instrucción para darle atención al que le sigue en orden.
* Función matarProceso(). Elimina al proceso activo de la lista de procesos en espera de ejecución, independientemente de si antes lo ejecutó o no. Este proceso eliminado será registrado en la cola *eliminados*.

De esta forma es que se logró la construcción del programa de forma exitosa, cumpliendo con todos los puntos solicitados. El trabajo remoto se realizó con mucha facilidad ya que utilizamos la plataforma **Github** para tener un repositorio del código, lo que nos permitió llevar un control más profesional del desarrollo del proyecto y las versiones del mismo. Se adjunta liga por si gusta verificar el contenido de este: [CesarYCG/Practica03\_SO (github.com)](https://github.com/CesarYCG/Practica03_SO)

Finalmente, una vez realizado y testeado el programa, se procedió a verificar si cumplía con los puntos previamente requeridos y una vez confirmado, se realizó el reporte escrito y el vídeo explicativo sobre su funcionamiento.

Conclusiones

**Calderón César.** Con el desarrollo de esta práctica pude entender de forma más detenida e interactiva cómo es que el SO crea, procesa, registra y controla los procesos creados y cómo es que se utiliza la información relevante de los mismos para llevar una mejor gestión de ellos, si bien no utilizamos conceptos como la prioridad o las formas de comunicación, si pude tener una aproximación más didáctica con la creación eliminación y ejecución de los procesos. Así mismo, el desarrollo del programa me hizo retomar mis habilidades de programación y desempolvarlos para poder cumplir con los puntos requeridos. En forma general encontré esta práctica bastante interesante y retroalimentativa.

**Maceda Fernando.** a