UNIVERSIDAD TECNOLOGICA DE SANTIAGO
(UTESA)

PRESENTADO POR:

ERIK CRUZ 1-18-0759

PRESENTADO A:

IVAN MENDOZA

ASIGNATURA:

COMPILADORES

ASIGNACION:

TAREA SEMANA 7

Santiago de los Caballeros
República Dominicana
Noviembre, 2023

Entorno de Ejecución

Un entorno de ejecución es un conjunto de instrucciones que proporciona los recursos esenciales y servicios necesarios para la ejecución de aplicaciones en un lenguaje de programación específico. Este entorno actúa como una capa de abstracción que separa el código de la aplicación del hardware y del sistema operativo subyacentes. Entre sus funciones principales se encuentran la gestión de la memoria, el manejo de excepciones y la prestación de servicios esenciales para garantizar la ejecución adecuada del código. Además, el entorno de ejecución puede incorporar un compilador o intérprete "justo a tiempo" (JIT) que traduce el código a instrucciones de máquina comprensibles y ejecutables por la computadora.

Es importante destacar que el entorno de ejecución es específico del lenguaje de programación y puede variar entre diferentes implementaciones del mismo.

Organización de la memoria en tiempo de ejecución.

Durante la ejecución de un programa, la memoria se organiza en diversas secciones, cada una con funciones específicas que contribuyen al funcionamiento eficiente y seguro del software. En el segmento de código, las instrucciones ejecutables se almacenan de forma generalmente inmutable. La sección de datos estáticos aloja variables globales e inicializadas que persisten a lo largo de toda la ejecución, permitiendo su acceso y modificación según las necesidades del programa. Por otro lado, el espacio BSS se reserva para variables globales no inicializadas, inicializándose dinámicamente durante la ejecución.

La pila, un componente dinámico esencial, se utiliza para gestionar la información local de las funciones y controlar las llamadas a funciones. Cada vez que se llama a una función, se crea un nuevo marco de pila que alberga los parámetros y variables locales, y la pila crece y decrece de manera dinámica conforme evoluciona la ejecución del programa. En contraste, el montón se reserva para el almacenamiento dinámico de datos, siendo utilizado para asignar y liberar memoria durante la ejecución, especialmente para estructuras de datos dinámicas como listas enlazadas.

La memoria compartida facilita la comunicación entre procesos concurrentes, proporcionando un espacio compartido donde los datos pueden ser accedidos y modificados por múltiples procesos. La memoria de código máquina almacena instrucciones compiladas para mejorar el rendimiento, mientras que la memoria virtual, gestionada por el sistema operativo, permite presentar la memoria física de manera continua a los programas mediante técnicas como la paginación y la segmentación.

Estrategias de asignación de memoria.

Mantiene un registro constante de todo el espacio libre en el montículo. Realiza dos funciones fundamentales: asignación y desasignación. En la asignación, proporciona memoria contigua del montículo del tamaño necesario, ya sea utilizando espacio libre existente o solicitando más al sistema operativo. En la desasignación, devuelve el espacio liberado a la reserva de espacio libre para su reutilización. La eficiencia en el espacio busca minimizar el uso total del montículo, reduciendo la fragmentación. La eficiencia del programa implica un uso hábil del subsistema de memoria para mejorar la velocidad de ejecución, considerando la "localidad" de los programas en la memoria. Además, se busca una baja sobrecarga, es decir, minimizar el tiempo dedicado a asignaciones y desasignaciones, especialmente para solicitudes pequeñas.

Acceso a variables locales, no locales y globales.

En un programa, el acceso a las variables locales se maneja a través de la pila, donde cada función tiene su propio marco de pila. Las variables locales se almacenan dentro de este marco, y su acceso y liberación están restringidos al tiempo de vida de la función. Este enfoque eficiente facilita la gestión de las variables en funciones, asegurando su correcta creación y destrucción conforme a la ejecución del programa.

Por otro lado, las variables no locales, también conocidas como variables en ámbitos superiores, incluyen parámetros de funciones o variables globales. El acceso a estas variables se realiza a través de referencias y registros de activación de pila, permitiendo que las funciones accedan a datos definidos en ámbitos superiores. Este mecanismo de acceso se basa en seguir la cadena de registros de activación de pila hasta llegar al ámbito adecuado, garantizando una gestión eficiente y coherente de las variables en distintos niveles de anidamiento de funciones.

Las variables globales, declaradas fuera de cualquier función, se almacenan en una sección específica de la memoria y se accede a ellas directamente mediante su nombre. Este tipo de variables está disponible para todo el programa, lo que facilita el intercambio de información entre diferentes partes del código, aunque puede introducir problemas relacionados con el encapsulamiento y el modularidad.

Paso de parámetros.

La forma en que los procedimientos se comunican entre sí se denomina paso de parámetros. Los valores de las variables de un procedimiento de llamada se transmiten al procedimiento llamado mediante algún mecanismo. Antes de continuar, es fundamental revisar algunos términos básicos relacionados con los valores en un programa.