Programación de Sistemas Depuradores (Debugger)

MIS. Lizbeth Hdz. Glz.

TIPOS DE ERRORES

- **Sintácticos .** Los errores de sintaxis, o sintácticos, ocurren cuando el programador escribe código que no va de acuerdo a las reglas de escritura del lenguaje de programación.

- **Lógicos.** Los errores lógicos ocurren a causa de un mal diseño del programa. Puede ocurrir que una línea de código observe todas las reglas sintácticas del lenguaje, pero el código tenga una lógica equivocada.

Depuradores

- Un sistema interactivo de depuración proporciona posibilidades al programador que le ayudan en la prueba y depuración de programas.

- Permite depurar o “limpiar” los errores de un programa.

Funciones y posibilidades de depuración

- Requisitos:
  - un conjunto de funciones de prueba
- Un grupo importante de dichas funciones se relaciona con la secuenciación de la ejecución, que es la observación y control del flujo de ejecución del programa.

el programador puede definir puntos de interrupción específicos:

- Una vez suspendida la ejecución, pueden usarse otros mandatos de depuración para analizar el avance del programa y diagnosticar los errores detectados;
- el programador puede definir expresiones condicionales que se evalúan continuamente durante la sesión de depuración.
- la ejecución del programa se suspende cuando cualquiera de esas condiciones se hace verdadera.

- Puede ser útil permitir que el programa se ejecute en varias velocidades llamadas pasos (gaits)
- El rastreo puede emplearse para seguir el flujo de la lógica de ejecución y las modificaciones de los datos.
- El flujo de control puede rastrearse con distintos niveles de detalle: módulo, subrutina, instrucciones de salto, etc.

- es importante que un sistema de depuración posea buenas posibilidades de despliegue del programa
- debe exhibir el programa que está depurando, junto con los números de proposición, y el usuario debe poder controlar el nivel en el que ocurra este despliegue, p.e.
  - exhibir el programa original
  - después de la expansión de macros,…

- También es de utilidad permitir modificar y re-compilar el programa incrementalmente (el sistema debe guardar todas las especificaciones de depuración)

Consideraciones del lenguaje

- Los mandatos para el depurador que inician acciones y recogen datos sobre la ejecución del programa deben ser comunes a todos los lenguajes;
- Un sistema de depuración también debe ser "sensible" al lenguaje específico que se está depurando.

Consecuencias

- Cuando el depurador recibe el control, se suspende temporalmente la ejecución del programa que se está depurando.
- El depurador, debe ser capaz de determinar el lenguaje en el que está escrito el programa y, por consiguiente, determinar su contexto;
- debe ser capaz de cambiar de contexto cuando un programa escrito en un lenguaje llame a un programa escrito en un lenguaje diferente.


Programa
---
llamadaOtro
depurador
Contexto
X
Contexto
Y

- las proposiciones de asignación que cambian los valores de variables durante la depuración deberán procesarse según la sintaxis, y semántica del lenguaje de programación fuente.
- La notación que se emplea para especificar ciertas funciones de depuración varía de acuerdo con el lenguaje del programa que se está depurando
- las funciones en sí se llevan a cabo de la misma forma
- el depurador debe tener acceso a la información reunida por el traductor

Optimización de código

- Muchas optimizaciones incluyen la readaptación de segmentos de código
  - suprimir las expresiones invariantes de los ciclos
  - expresiones redundantes pueden eliminarse (pueden desaparecer proposiciones completas)
  - bloques de código pueden reordenarse para eliminar instrucciones de salto innecesarias

Problemas de optimización

- El reordenamiento del código altera la secuencia de ejecución y puede afectar al rastreo (trace),
- Puede ser difícil relacionar un error con su origen
- La depuración de código optimizado requiere una cantidad sustancial de cooperación del compilador optimizador

- Algunas optimizaciones más complejas no pueden manejarse con tanta facilidad,
- el depurador tan sólo debe informar al usuario de que no está disponible una función particular VS intentar alguna imitación incompleta de la función.

Relación con otras partes del sistema

- Un depurador interactivo debe estar siempre disponible
- debe aparecer al menos como parte del ambiente al momento de ejecución y como parte integral del sistema
- muchos fallos de los programas no pueden repetirse fuera del ambiente de producción

- El depurador también debe existir de una forma consistente con la seguridad e integridad de los componentes del sistema. Nadie debe poder usar el depurador para acceder a algún dato o código que de otra forma seria inaccesible

Dump

- un vaciado puede incluir información que se ha dejado en el almacenamiento, el depurador presenta sólo información del contenido de objetos específicos con nombre.
- El depurador debe coordinar sus actividades con las de compiladores e intérpretes de lenguajes actuales y futuros

- El lenguaje de mandatos debe tener una sintaxis clara, lógica y simple.
- También debe ser lo más parecido posible a los lenguajes de programación.
- Los mandatos deben ser simples, en vez de compuestos, y con la menor cantidad posible de parámetros
- Que los parámetros no tengan errores en atributos como el tipo y la gama de valores; también se les debe asignar valores por omisión a la mayoría de ellos.

- debe minimizar el uso de símbolos de puntuación como paréntesis, diagonales, comillas y otros caracteres especiales
- Cualquier buen sistema interactivo debe tener un dispositivo de AYUDA en línea

Resumen

- El depurador permite detener el programa en:
  - Un punto determinado mediante un punto de ruptura (breakpoint).
  - Un punto determinado bajo ciertas condiciones mediante un punto de ruptura condicional.
  - Un momento determinado cuando se cumplan ciertas condiciones.
  - Un momento determinado a petición del usuario.

Durante esa interrupción, el usuario puede:

- Examinar y modificar la memoria y las variables del programa.
- Examinar el contenido de los registros del procesador.
- Examinar la pila de llamadas que han desembocado en la situación actual.
- Cambiar el punto de ejecución, de manera que el programa continúe su ejecución en un punto diferente al punto en el que fue detenido.
- Ejecutar instrucción a instrucción.
- Ejecutar partes determinadas del código, como el interior de una función, o el resto de código antes de salir de una función.

EJEMPLOS DE DEPURADORES:

- **SoftICE** es una herramienta excepcionalmente útil para el desarrollo de drivers y es compatible con las últimas versiones del sistema operativo de Microsoft.
- **OllyDbg.** Funciona a nivel de ensamblador de 32 bits y es especialmente útil cuando no se dispone del código fuente.
- **GNU Debugger (gdb)**
- **Trw2000**
- **Ida Pro**
- **Cheat Engine**