Universidad de Guadalajara

Ingeniería en computación

Computación Tolerante a Fallas 2023ª

“Herramientas para el manejo de errores”

Olguín Hernández Jair Benjamín.

217439707

Durante el proceso de desarrollo podemos tener muchos errores, ya sean errores que hagan que el código no compile correctamente o el funcionamiento sea lo contrario a lo esperado. Pero aún más importante son los errores que afectan al usuario, cuando se trata de agregar una nueva funcionalidad y esta afecta a las funciones que ya estaban en uso. Haciendo que el usuario no tenga una experiencia adecuada, por lo que debemos tener herramientas que nos ayuden a evitar o limitar estas situaciones. Para manejar errores en programación, es importante contar con una serie de herramientas y técnicas que permitan detectar, identificar y corregir los problemas en el código. Algunas de estas herramientas:

1. Depuradores: Un depurador es una herramienta que permite seguir el flujo de ejecución de un programa, detenerlo en puntos específicos, inspeccionar y modificar variables, entre otras funciones. Los depuradores más comunes son gdb (para C/C++), lldb (para Swift), y pdb (para Python).
2. Assertions: Las assertions son declaraciones que indican que una condición dada debe ser verdadera. Si una assertion falla, se genera una excepción. Los lenguajes de programación modernos suelen incluir una palabra clave de assertion, como assert en Python, o NSCAssert en Objective-C.
3. Manejo de excepciones: El manejo de excepciones es una forma de controlar errores en tiempo de ejecución. En lenguajes de programación orientados a objetos, se utilizan para controlar situaciones excepcionales mediante un bloque de código específico. Por ejemplo, en Java o C# se utiliza try-catch.
4. Test Unitarios: Los test unitarios son una técnica para probar pequeñas porciones de código de forma aislada, lo que permite detectar errores en una fase temprana del desarrollo. Existen varias herramientas para automatizar la creación y ejecución de test unitarios, como JUnit para Java, NUnit para .NET, o unittest para Python.

5. Registro de errores: Tener un registro de los errores que ocurren en un sistema es crucial para poder investigar y solucionar problemas. Herramientas como el registro de eventos de Windows o el sistema de registro de Linux son útiles para registrar errores de sistema, mientras que un framework de registro de aplicaciones como log4j o NLog pueden ser utilizados para registrar errores en aplicaciones.

Un test unitario es un tipo de prueba automatizada que se utiliza para comprobar que una pequeña porción de código, conocida como unidad, funciona correctamente. Estas pruebas son escritas por desarrolladores de software y se ejecutan en una fase temprana del ciclo de desarrollo, antes de que el código sea liberado a producción. Un test unitario debe cumplir con ciertas características:

- Debe ser fácil de escribir y ejecutar.
- Debe ser independiente entre sí, es decir, no debe depender de otros test para funcionar.
- Debe ser rápido.
- Debe ser específico.
- Debe ser automatizado.
- Debe ser mantenible.

Los test unitarios se utilizan para detectar problemas temprano en el desarrollo, reduciendo así el tiempo y costo de corregir los errores. Esto también ayuda a garantizar que los cambios realizados en el código no afecten negativamente a otras partes del sistema.

Ejemplo:

```python
import unittest
def suma(a, b):
    return a + b

class TestSuma(unittest.TestCase):
    def test_suma_positiva(self):
        result = suma(1, 2)
        self.assertEqual(result, 3)

    def test_suma_negativa(self):
        result = suma(-1, -2)
        self.assertEqual(result, -3)

    def test_suma_cero(self):
        result = suma(0, 0)
        self.assertEqual(result, 0)

if __name__ == '__main__':
    unittest.main()
```

En este ejemplo, creamos una función llamada suma() que toma dos argumentos y devuelve su suma. Luego, creamos una clase llamada TestSuma que hereda de unittest.TestCase y contiene tres métodos de prueba, test_suma_positiva(), test_suma_negativa(), y test_suma_cero(). Cada uno de estos métodos utiliza el método assertEqual() para comprobar si el resultado de la llamada a suma() es igual al valor esperado. Por último, si ejecutas el archivo python, las pruebas unitarias se ejecutarán automáticamente y te darán un reporte de los resultados de estas.

Conclusión:

En conclusión, el manejo de errores es esencial en el desarrollo de software ya que permite detectar y corregir problemas de manera temprana y eficiente. Esto ayuda a garantizar la calidad del producto final y a evitar problemas en producción. Además, el uso de pruebas unitarias y la implementación de manejadores de excepciones puede ayudar a garantizar que el código sea robusto y confiable. Es importante también tener una buena documentación de los errores conocidos y las soluciones implementadas para poder tener un mejor control de estos.