UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS132 - Imperative Programming

Valgrind Errors

Contents

1 InvalidFree

Explanation: Trying to free (delete) memory that was *not allocated with* `new` or was *already freed.*

```
int main() {
    int x = 10;
    delete &x;  // Invalid: x was not allocated with 'new'
    return 0;
}
```

2 UninitCondition

Explanation: Using a variable in a condition before giving it a value.

```
int main() {
    int x;
    if (x > 0) {  // x is uninitialized
        // Do something
    }
    return 0;
}
```

3 InvalidRead

Explanation: Trying to read memory that doesn't belong to your program, such as reading past the end of an array.

```
int main() {
    int* arr = new int[5];
    int val = arr[10];  // Invalid read: index 10 is out of bounds
    delete[] arr;
    return 0;
}
```

4 UninitValue

Explanation: Using a variable without initializing it first, even outside of a condition.

```
int main() {
    int x;
    int y = x + 5;  // x is uninitialized
    return 0;
}
```

5 SIGFPE (Floating Point Exception)

Explanation: Happens when dividing by zero.

```
int main() {
    int x = 5;
    int y = 0;
    int z = x / y;  // Division by zero
    return 0;
}
```

6 InvalidWrite

Explanation: Writing to memory you shouldn't, like past the end of an array.

```
int main() {
    int* arr = new int[5];
    arr[10] = 42;  // Invalid write: index 10 is out of bounds
    delete[] arr;
    return 0;
}
```

7 SIGSEGV (Segmentation Fault)

Explanation: Accessing memory that's not allowed (e.g., null or invalid pointer).

```
int main() {
    int* ptr = nullptr;
    *ptr = 5;  // SIGSEGV: dereferencing a null pointer
    return 0;
}
```

8 MismatchedFree (Simplified)

Explanation: Using `delete` for memory allocated with `new[]` instead of `delete[]`.

Correct:

```
int* arr = new int[5];
delete[] arr;  // Correct way to free an array
```

Incorrect:

```
int main() {
    int* arr = new int[5];
    delete arr;  // Mismatched free: should use delete[]
    return 0;
}
```

9 FishyValue

Explanation: Pointer has a strange or clearly invalid value like `0x1`.

```
int main() {
    int* ptr = (int*)1;  // Fishy value: 1 is not valid memory
    *ptr = 42;  // Will likely crash
    return 0;
}
```