Call Stack and Call Hierarchy

1. Call Hierarchy

Call hierarchy describes the logical order of function calls in a program. It shows which function calls which other function, forming a tree structure.

Example:

```
void C() { }

void B() { C(); }

void A() { B(); }

int main() { A(); }
```

Call hierarchy: main() → A() → B() → C()

2. Call Stack

The call stack is a runtime memory structure that stores active function calls. It follows the Last In, First Out (LIFO) principle.

What the Call Stack Stores:

1. Function parameters
2. Local variables
3. Return address

Stack Example:

```
| C |
| B |
| A |
| main |
```

Call Stack and Debugging

When a program crashes, debuggers show a stack trace. It lists the function calls that led to the error.

Example Stack Trace:

```
#0 C()
#1 B()
#2 A()
#3 main()
```

Call Stack and Recursion

Each recursive call pushes a new frame onto the stack. Without a base condition, this leads to stack overflow.

Invalid Example:

```
void f() { f(); }
```

Summary

1. Call hierarchy shows function relationships.

2 Call stack manages execution at runtime.

3 Stack uses LIFO ordering.