FreeRTOS Architecture Part 1

Name

Universidad Panamericana

Presentation August 17, 2024

Contents

Why Cpp instead of just C?

Coding standards

Good code practices with clang-tidy and cpplint

Automated testing (Unit Test) and manual testing

Version control process GIT

Benefits of integrating CI into the software development lifecycle

Understanding Continuous Integration

Defensive Programmig

Defensive Programmig. Expect the unexpected

Defensive programming is a bit like always wearing a full suit of armor. It's about preparing for the worst while hoping for the best, much like someone living in a zombie apocalypse with a bunker full of canned goods. In this approach, every function call is a potential trick, every user input a Trojan horse, and paranoia isn't just recommended, it's required!

Defensive Programmig. Expect the unexpected

Good Practice

Defensive Programmig. Expect the unexpected

- Thats why is a good practice to use C++
- C++ has Zero Over Head Principle.
- Learn C++ and you can find a better job,and just add a plus to your resume.

Using const for Safety

- const keyword: ensures variables are not modified after initialization.
- Use const to protect function parameters, class members, and pointers.
- Example: void process(const Data& data); guarantees data remains unchanged.
- This is used for read only variables.

Using const in C++

```
    class Person {
      public:
          string name;
          int age;
          Person(string n, int a) : name(n), age(a) {}

          void print() const {
              cout << "Name: " << name << ", Age: " << age
                  << endl;
          }
      };

      void displayPerson(const Person& p) {
          p.print();
      }
```

Memory Managment

Memory Hierarchy: A Light-Hearted Tour

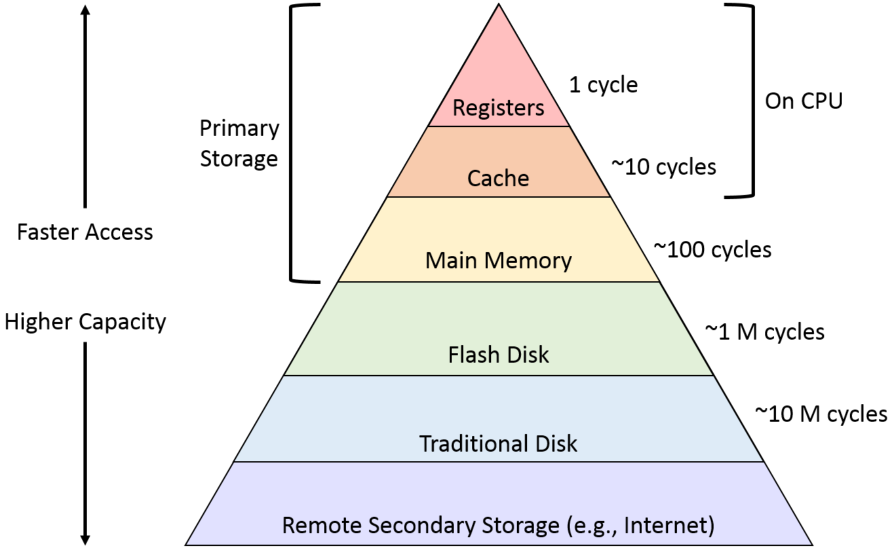
- **Registers:** The speed-demons of memory. Too fast to care, but you really should!
- **Cache:** The backseat driver of computing. It makes decisions you didn't ask for, often with surprising results.

Friendly Reminder

Regularly clearing your cache: not just good practice, it's like digital detox for your devices!

- **RAM (Random Access Memory):** The workaholic of memory. When it runs out, things go south quickly—plan wisely!
- **Storage:** The elephant's graveyard. Where all your code and files go to rest. Yes, your code lives somewhere physical!

Memory Hierarchy



The Memory Hierarchy

How does many values has singles variable?

- One?
- Two?

How does many values has singles variable?

- One?
- Two?

A variable has two values

- One : Its current value
- Two : Its current addres

Passing by Copy

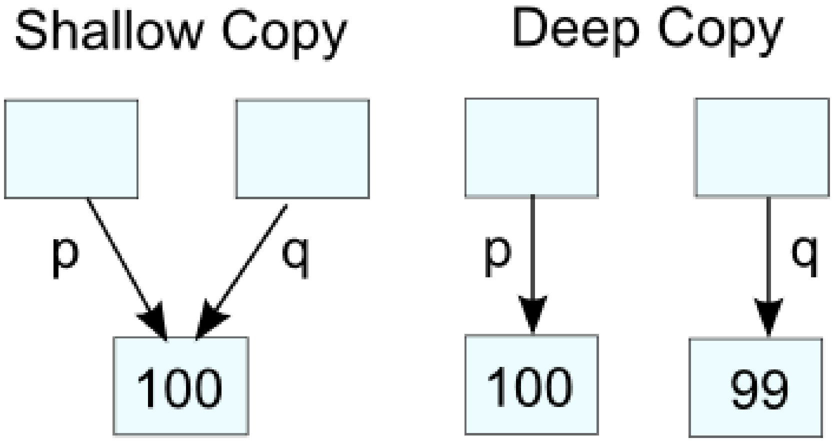
- When parameters are passed by copy, a new instance of the argument is created.
- Modifications within the function do not affect the original variable.
- Best used when you need to ensure the original data remains unchanged.

```
void incrementByCopy(int x) {
    x = x + 1;
    cout << "Inside function: " << x << endl;
}

int main() {
    int a = 5;
    incrementByCopy(a);
    cout << "Outside function: " << a << endl;
}
```

Passing by Reference

- Passing by reference sends a reference to the original variable.
- Any changes inside the function affect the original variable.
- More efficient for large data structures but must be used carefully.

```cpp
void incrementByReference(int& x) {
    x = x + 1;
    cout << "Inside function: " << x << endl;
}

int main() {
    int a = 5;
    incrementByReference(a);
    cout << "Outside function: " << a << endl;
}
```


Shallow Copy
p
q
100
Deep Copy
p
100
q
99