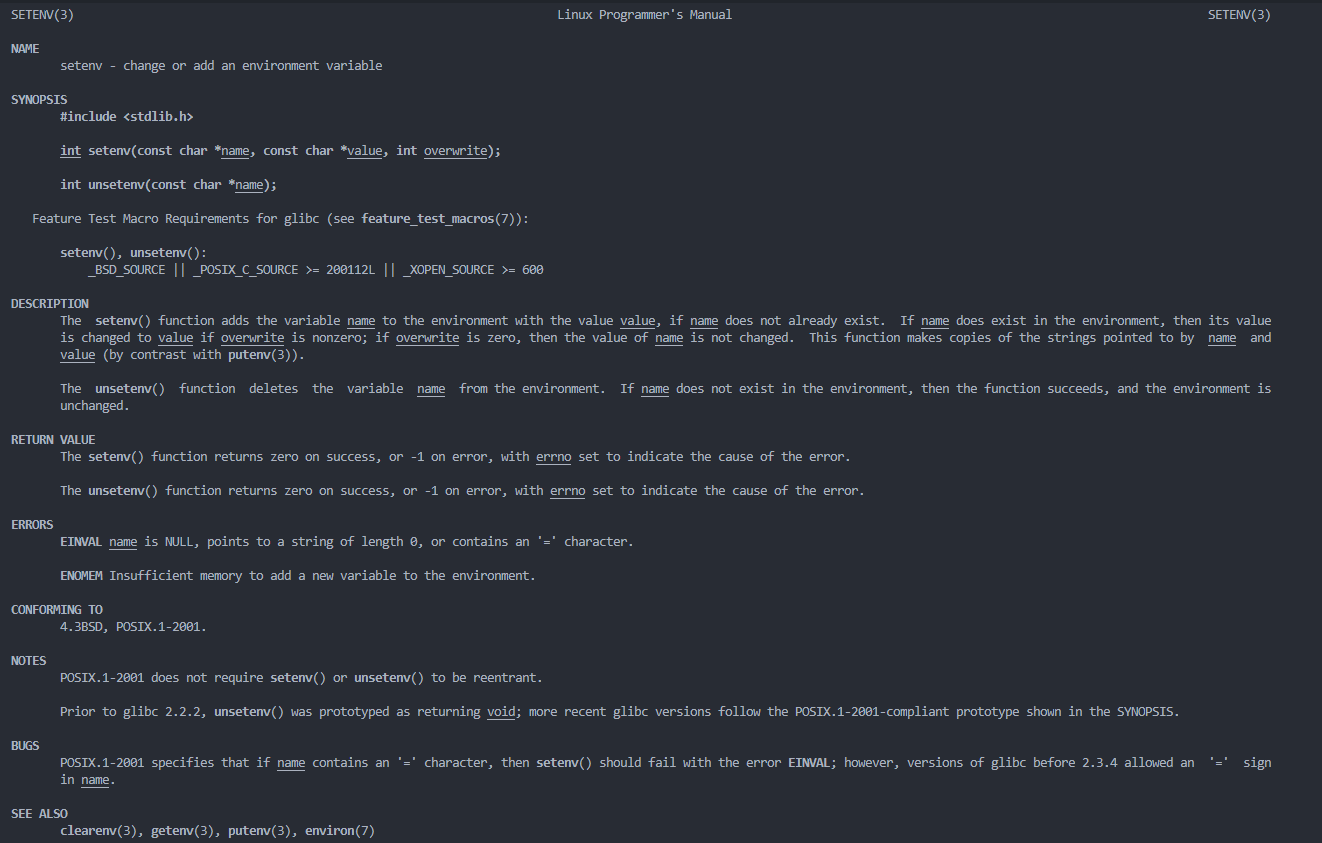
1 开发过程中用到的一些Linux系统调用和一些C++函数

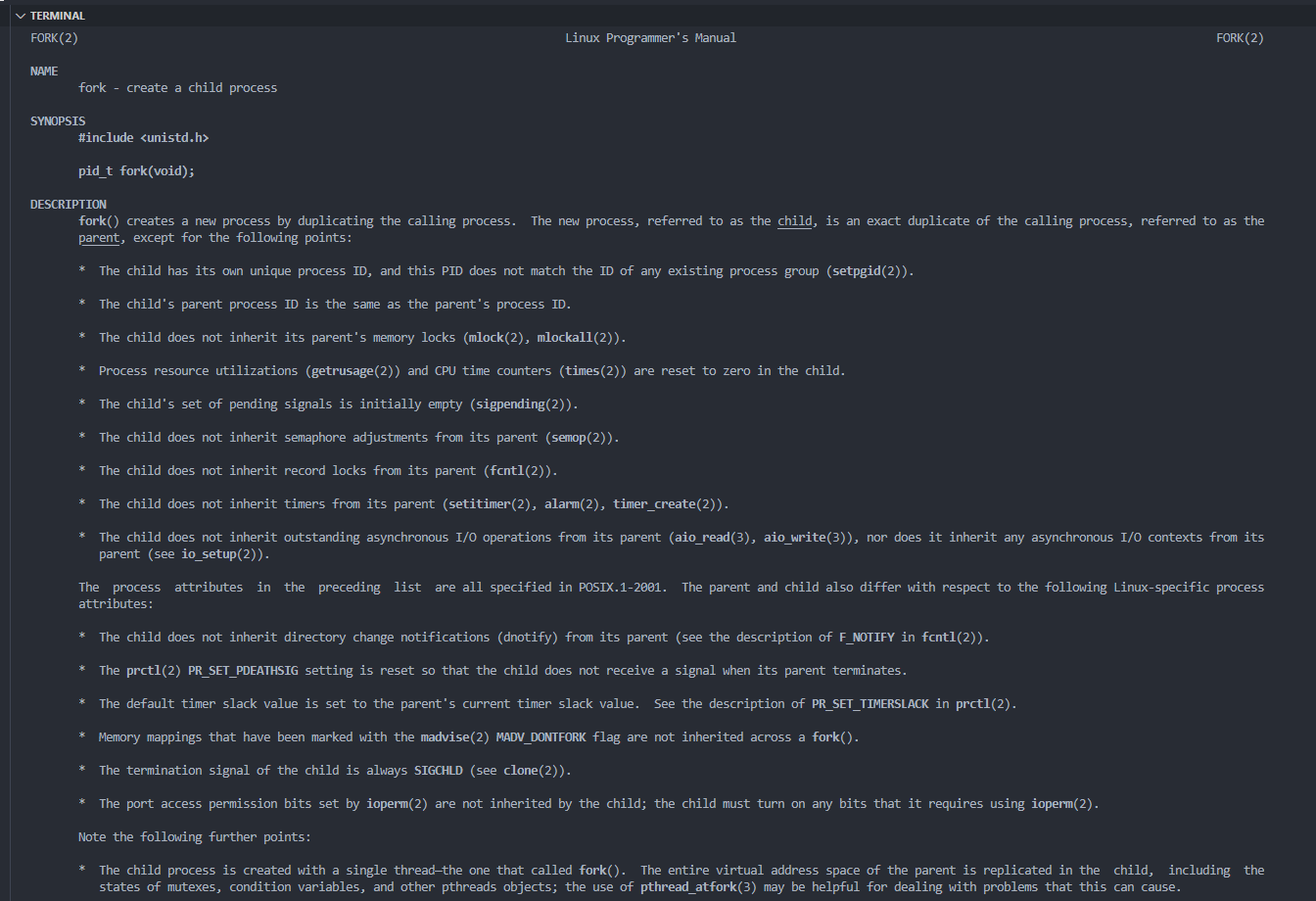
Chdir



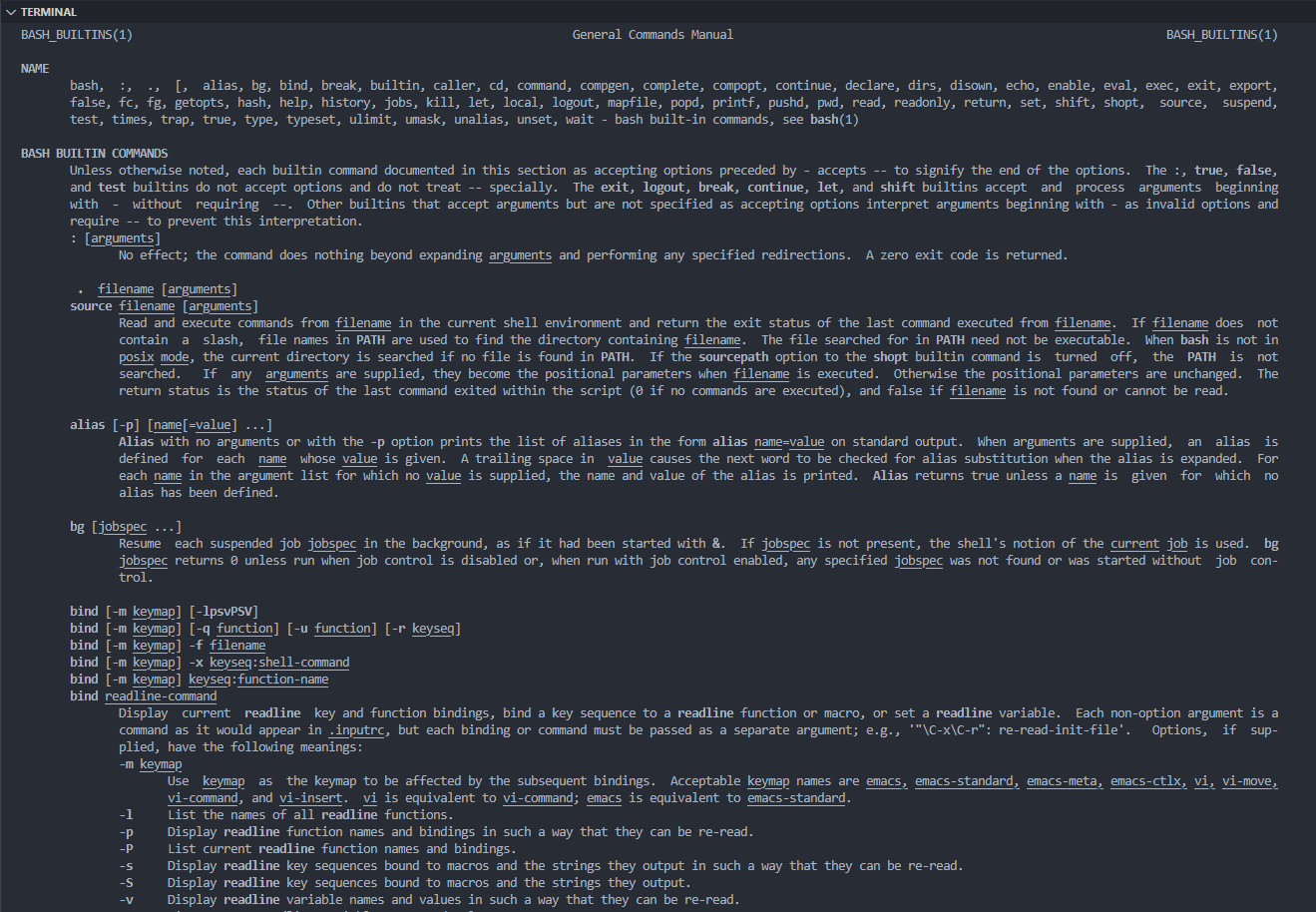
Setenv



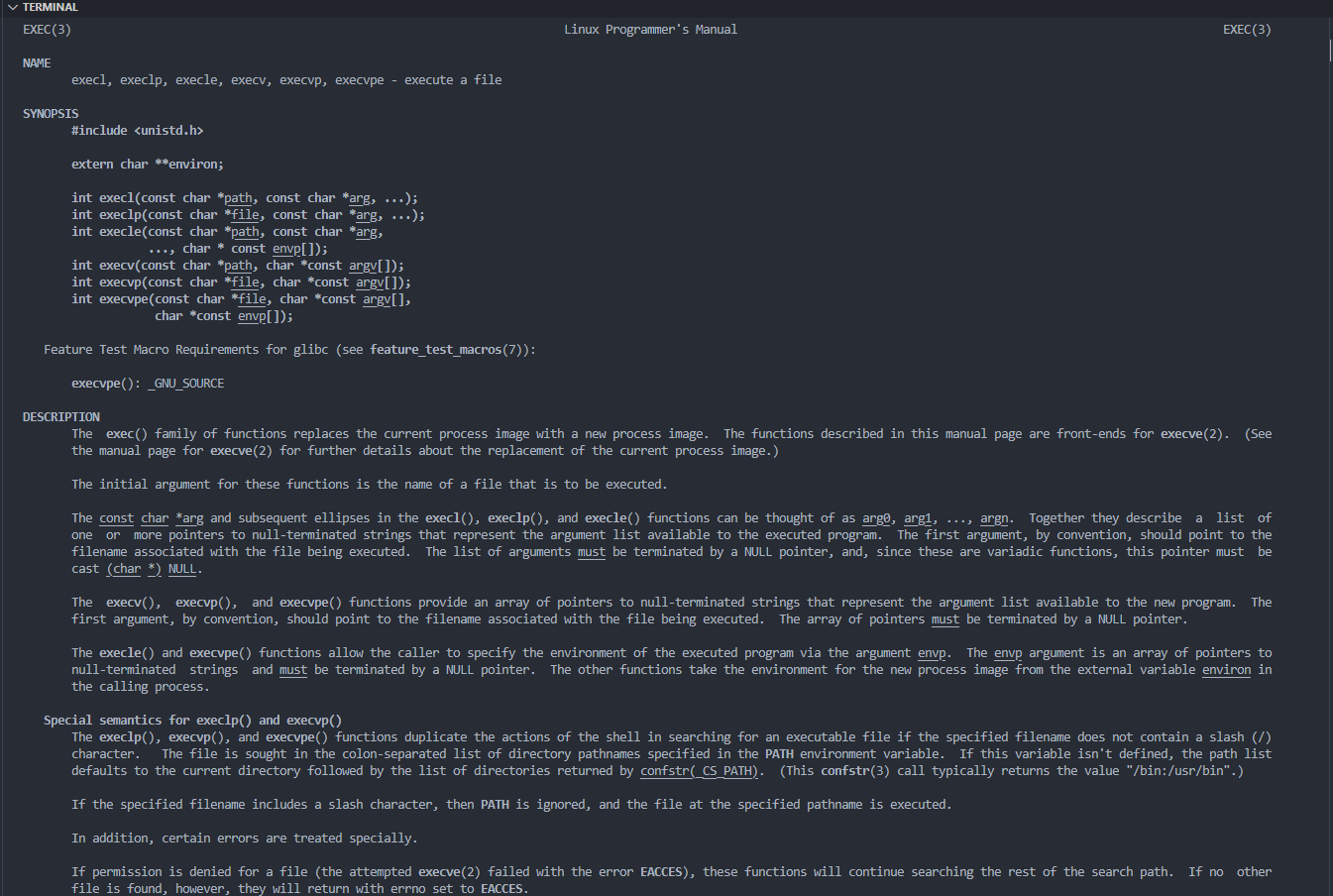
Fork



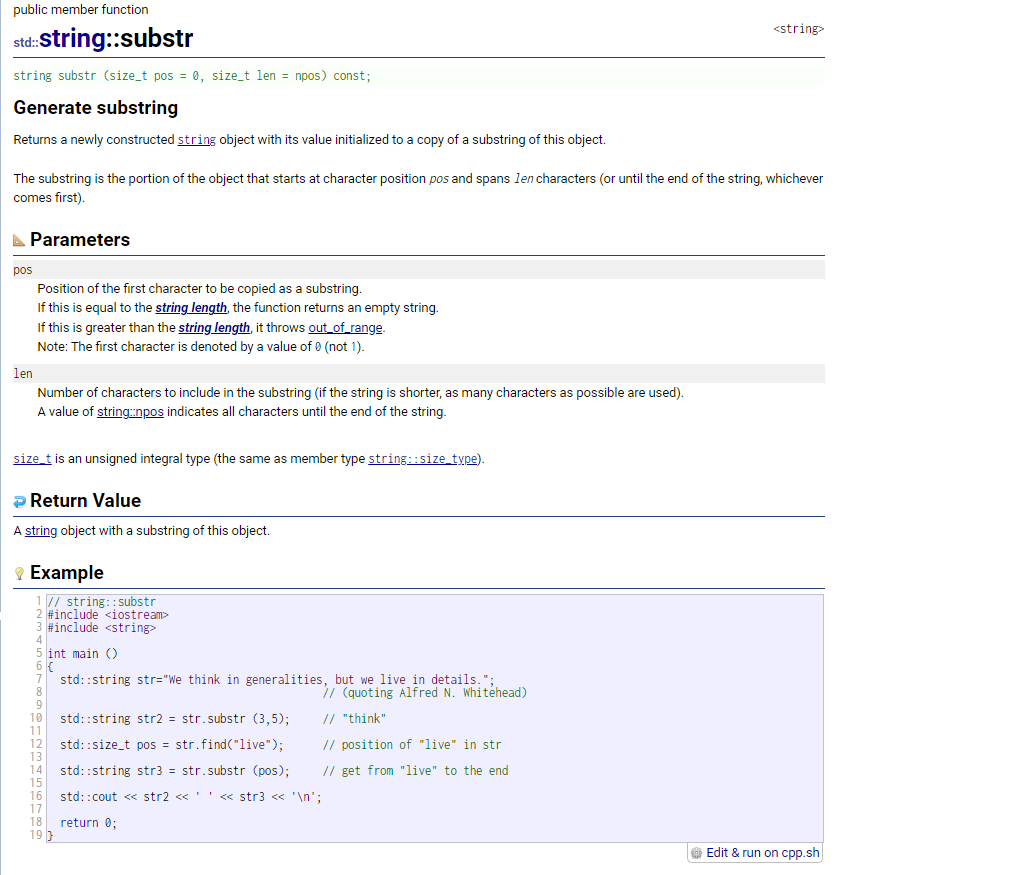
wait



execvp

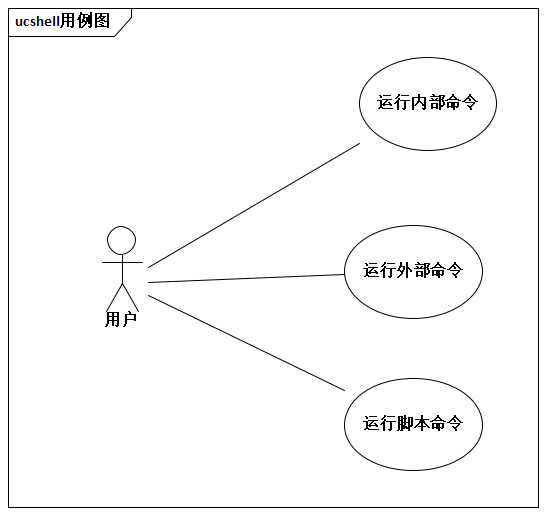


Substr

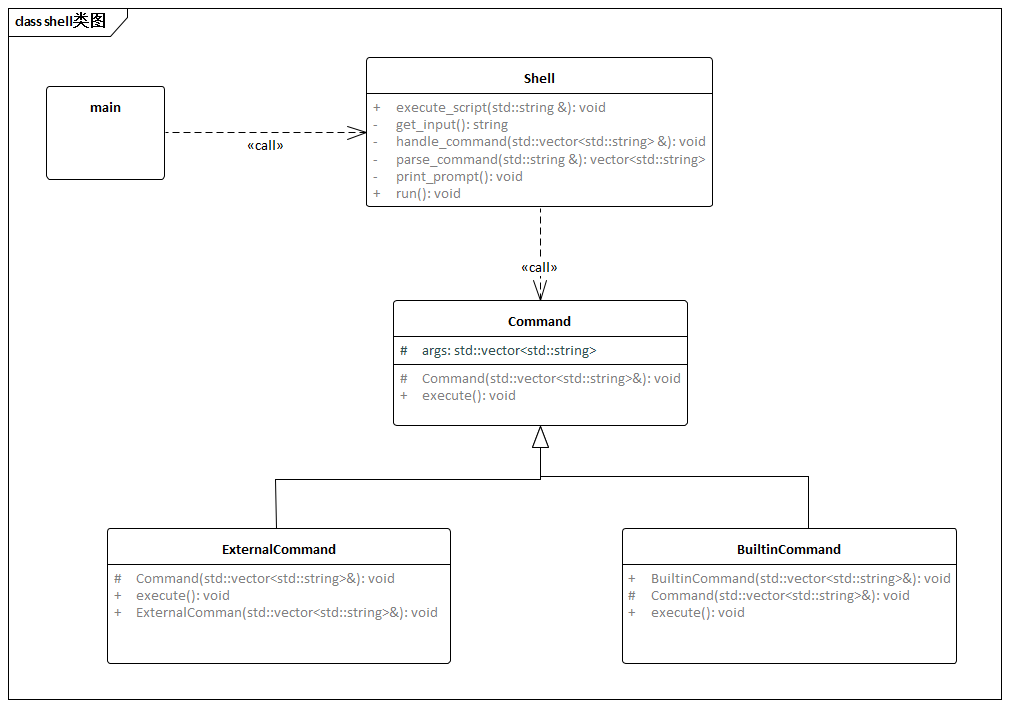


2 UML图

用例图



类图



这段代码展示了三个类的定义和它们之间的关系：

Command 类：

Command 是一个抽象基类，它定义了执行命令的接口 execute()，并且持有命令执行所需的参数 args。.Command 类中的 execute() 方法是纯虚函数，需要在派生类中进行实现。

BuiltinCommand 类：

BuiltinCommand 是 Command 的派生类，表示内置命令。

BuiltinCommand 接受一个 std::vector&lt;std::string&gt; 类型的参数 args，并将其传递给基类 Command 的构造函数进行初始化。

BuiltinCommand 类重写了 execute() 方法，用于执行特定的内置命令逻辑。

ExternalCommand 类：

.ExternalCommand 也是 Command 的派生类，表示外部命令。

类似于 BuiltinCommand，ExternalCommand 也接受一个 std::vector&lt;std::string&gt; 类型的参数 args，并将其传递给基类 Command 的构造函数进行初始化。

ExternalCommand 类同样重写了 execute() 方法，用于执行外部命令的逻辑。

关系

继承关系：

BuiltinCommand 和 ExternalCommand 都是 Command 的子类，它们继承了 Command 类的接口和数据成员。

共享接口：

所有三个类都实现了 execute() 方法，但具体的执行逻辑在每个类中可能有所不同，根据命令的类型（内置命令或外部命令）来进行特定的处理。

命令类型的区分：

BuiltinCommand 和 ExternalCommand 之间的主要区别在于它们处理的命令类型不同。内置命令可能直接由 Shell 执行或者由 Shell 本身提供逻辑支持，而外部命令则是通过调用系统的外部程序实现。

这种设计模式允许系统灵活地处理不同类型的命令，并且通过基类和派生类的关系，实现了代码的重用和扩展性。

活动图

