关于Nodejs多进程与child_process模块的研究

2020年1月30日 14:31

基本模块Child_Process模块

分类:

1. 异步模型
   通过提交给nodejs引擎创建进程的事务，于当前代码执行结束后创建进程，通过事件回调来响应和操作进程

   - Asynchronous Process Creation
     - Spawning `.bat` and `.cmd` files on Windows
     - `child_process.exec(command[, options][, callback])`
     - `child_process.execFile(file[, args][, options][, callback])`
     - `child_process.fork(modulePath[, args][, options])`
     - `child_process.spawn(command[, args][, options])`
       - `options.detached`
       - `options.stdio`

2. 同步模型
   执行后立即创建进程，进程执行完成后返回进程执行的输出内容，函数与异步模型基本对应，后缀Sync

   - Synchronous Process Creation
     - `child_process.execFileSync(file[, args][, options])`
     - `child_process.execSync(command[, options])`
     - `child_process.spawnSync(command[, args][, options])`

主要函数

- fork函数。用来执行nodejs模块，或者说直接执行js文件，直接提供nodejs文件路径而不需要"node path"方式执行,**只有异步版本**
- exec函数，产生一个子shell，并直接执行一个shell命令，与在shell中手动执行命令相同，**参数直接包含在command字符串中，易于使用**
- execFile函数，用于执行可执行文件，**而非shell命令，**其余与exec相同
- spawn函数，底层函数，执行一个子进程，并返回其流接口（stdio)

层次关系

- 底层：spawn函数，返回流对象，不缓冲输入输出
- 上层：exec execFile函数，**基于spawn函数实现，缓冲子进程的标准输出**
- 最上层:fork函数，**只能用于nodejs程序之间，**原因为其约定了基于node的特殊通信通道，通过process.on 和send函数，以消息形式发送信息，而非通过标准的输入输出流

标准对象

ChildProcess对象

- 表示一个进程，所有异步函数（exec fork spawn execFile）都返回一个ChildProcess对象
- ChildProcess对象中有操作子进程的基本方法例如关闭等，及其标准输入输出流包括：**stdout stdin stderr**

Buffer对象

- 同步函数缓冲**子进程的输出**并得到一个Buffer对象，可转换到string

主要问题解答

1. 同步函数返回值：Buffer对象包含了子进程的所有输出，因而其缓存子进程输出
2. 异步函数返回值：

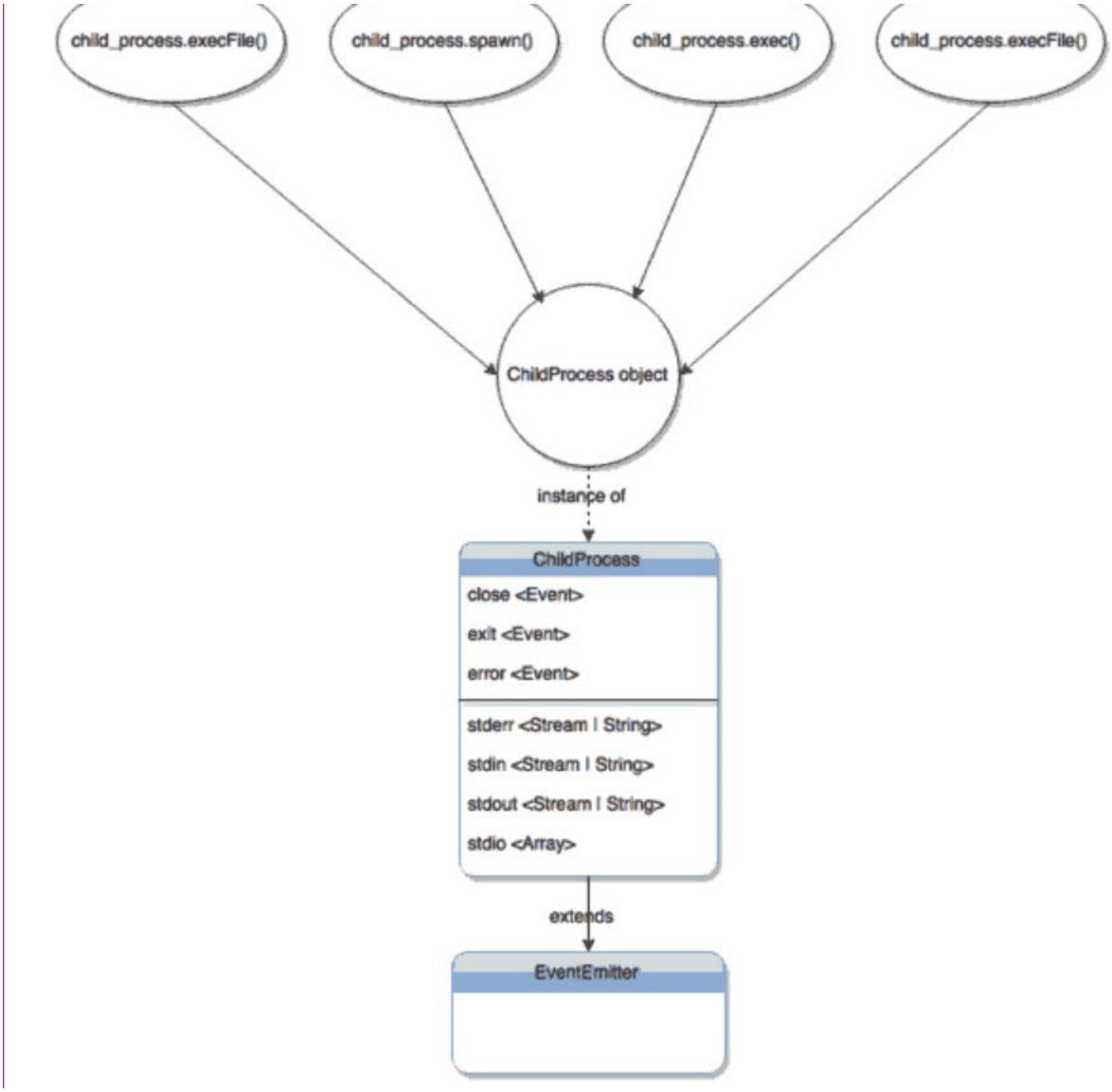
- exec和execFile函数，直接返回ChildProcess，允许通过流方式操控，同时在执行结束后调用回调函数（如果有），提供记录的子进程输出内容，因而其会缓存子进程输出
- spawn函数不缓冲子进程输出，没有回调函数，直接返回ChildProcess，只能通过流的方式操控子进程，读取输出通过stdio
- fork函数没有回调函数，返回ChildProcess对象，可通过流方式控制，同时**提供专用消息通道用于子node程序与host程序的通信**

主要操作方式

1. 通信：

- fork启动的node程序可通过ChildProcess和process对象中的on和send函数通信
- 其他方式启动的程序只能通过ChildProcess中的stdout和stdio通信，主要通过其 on函数接收数据，通过write pipe emit函数写入数据

架构图



注意

在Typescript中spawn函数有各种类型的返回值，其可以认为**是模板特化的结果,**实际只有一个ChildProcess类