# Proj2过程分析

本过程主要是分析sh.coff的执行过程，并不面面俱到，也并不全是正确的。

1. 首先，我们必须在配置文件中说明我们要执行的文件（sh.coff）。
2. process.execute:加载可执行文件，然后新建一个线程来执行。

这个步骤分为：加载（load）、执行（fork）。

## 加载

1.1加载的第一步是open，意思就是打开这个可执行文件（sh.coff）。

1.2将这个文件转化为Coff类（new），实质上就是设置两个变量：此程序第一条指令的虚拟地址（entryPoint）和可执行文件的片段（sections）。

注：相信大家以前都对可执行文件的格式了解甚少，这个对照着格式来看就很好理解。

简单点说，可执行文件包含几个部分：文件头、可选头、节等等，我们需要找到某些信息（具体怎么找先不管），最后我们得到entryPoint为1232，和4个sections（.text|.rdata|.data|.bss）

1.3计算需要的页数（numPages），4个sections所需要的页（）+程序的栈所需的8页+参数所需要的1页。同时赋值了程序计数器initialPC（程序入口）和栈指针initialSP。

1.4loadSections:为程序分配内存并加载coff片段（4个sections）进入内存。

分配内存是在内核管理的空闲页队列里面取相应数量的页面，将其存储在用户线程的页表中，这里要理解的是内核管理的空间和用户管理的空间的不同。而且空闲页结构中存在虚拟地址和物理地址的映射关系（这个我理解不一定对）。

1.5在最后一页里存储参数

注：在加载过程中，系统就将所有的页分配给了用户进程，而且将用户进程代码段、初始化数据段等程序运行所需的一切信息加载到物理内存中，是一次性的加载。

## 执行

这个执行时一个完全模拟CPU指令执行的过程，而对于指令的细节，我并不是很理解。我大概知道这个coff执行文件里面就是一条条CPU能读懂的指令，然后我们的模拟CPU不停地去取指令，翻译指令，执行指令。这里的最终体现就是一个个系统调用。

系统调用在执行之前，程序会将各个寄存器的值给设置好，然后就可以通过寄存器里面的值找到系统调用号和系统调用的参数，最终找到对应处理程序。

补充：这个执行过程除了体现在系统调用，还体现在对指令的异常处理上，这个异常处理是虚拟内存的内容。

通过OneInstruction(instr)模拟mips，将一条指令进行分割，并软件模拟执行。

其中，在OneInstruction函数中，通过readMem，读取主存中当前PC值指向的地址里的指令。

在readMem函数中，通过translate函数对传入的虚地址做转换。

在translate函数中，如果虚地址没有找到对应的实地址转换，就会抛出异常MipsException。这个MipsException将被捕获，进去异常处理（handle）流程中。

最终调用ExceptionHandler函数对不同的异常做相应的处理。

注：这里的异常包括缺页中断、系统调用......