PA3 必答题

请结合代码解释仙剑奇侠传，库函数, libos, Nanos-lite, AM, NEMU 是如何相互协助，来分别完成游戏存档的读取和屏幕的更新.

总的来说，仙剑奇侠传和库函数以及 libos 的代码最终形成 img，Nanos-lite 这个微量级的操作系统来对 img 进行加载以及进行上下文的切换、不同程序的切换等工作，AM 用来模拟一个运行时的环境：时钟、输入输出等，NEMU 用来提供程序的运行时环境。在程序运行之前，通过我们的编译等工作，已经将仙剑奇侠传的代码以及库函数编译链接等形成了 img 文件。然后我们的 NEMU 的主程序加载微量级的操作系统的镜像，然后操作系统开始初始化，然后读取程序的 img，然后跳到程序的入口执行程序。其中 NEMU 执行一条指令的过程就是我们 PA2 实现的过程，他其实是模仿一条 CPU 取值、译码以及执行的过程，当执行的过程中遇到了 system 的指令的时候，就会进行上下文的切换，然后跳到我们 PA3 实现的文件系统调用以及其他的系统调用的部分，最后再调回系统调用后的下一条指令，最终完成游戏的执行。

游戏存档的读取过程如下：首先执行语句 fp = fopen(szFileName, "rb")，这条语句的 fopen 对应的函数体就是在库函数中进行定义的，然后 NEMU 执行 fopen 对应的 riscv32 的汇编文件的时候，就会调用 SYS_open 的系统调用，这个系统调用就是执行我们的文件系统中的 fs_open 的函数，在调用的过程之中，我们的微量级的操作系统就提供了上下文切换的功能，然后 fread 和 fclose 等过程的执行都与 fopen 类似，只是对应的系统调用不同，最终执行完所有的文件相关的函数之后，在调用 memcpy 这个库函数汇编之后的代码来执行，最终完成游戏的读档。

屏幕更新的过程如下：NDL_DrawRect()这个函数也是在库函数之中定义的，他也是在编译的过程之中就被写入了镜像文件，然后 NEMU 执行到这个部分的时候就跳转到该库函数对应的代码段，之后在 DrawRect 函数中遇到了向 canvas 写入的操作，在之前的 PA2 定义的 AM 之中，我们对 canvas 内存空间的写入会调用相应的 callback 函数，这个 callback 函数就起到了同步屏幕内容的功能，最终完成了屏幕的更新。