Lab6挑战性任务

在Lab6基础部分的实验中我们实现了一个简单的shell。在本次挑战性任务，我们将通过实现一些难度层次递进的小组件或附加功能，来丰富我们的shell，从而加深对整个MOS lab6的理解，让我们的MOS更加完整。

> 前置要求：实现lab5 fsformat中"文件夹生成"的相关代码。

Easy部分

(1) 实现后台运行

在一个命令后增加 & 符号，使得shell不需要等待此命令执行完毕后再继续执行，而是将此命令置于后台执行，此时可在shell继续输入新命令。

> 提示：目前的shell在等待输入时，是在内核态忙等，需要对此进行修改才能实现后台运行。

(2) 实现一行多命令

用 ; 分开同一行内的两条命令。

> 提示：我们保留symbol里已经预留有 ';' 和 '&' 字符。

(3) 实现引号支持：

实现引号支持后，shell可以处理如：`echo.b "xxx | xxx"` 这样的指令。完成这项任务时，可以仅实现强引用。

(4) 实现如下命令

- tree
- mkdir
- touch

(5) 实现请屏

可以通过监听Ctrl+L或者实现一个clear内建指令完成这个任务

> 提示：可以参考linux如何实现请屏（输出特殊字符控制终端光标）

(6) 尝试彩色输出

通过putty/secureCRT等终端工具+ANSI控制序列，可以控制彩色输出。

如果你对这部分不是很了解，建议上网搜索一下"字符编码 ANSI"。或者你可以打开任何一个Lab的评测Log，观察"彩色输出部分"是如何实现的

Normal部分——历史命令功能

任务背景

在linux下我们输入的shell命令都会被保存起来，并可以通过up/down键回溯指令，这为我们的shell操作带来了极大的方便。

任务目标

实现保存在shell输入的指令，并可以通过history.b命令输出所有的历史指令以及通过上下键回溯指令。

任务要求

第一部分：要求我们将在shell中输入的每步指令，在解析前/后保存进一个专用文件（如.history）中，每行一条指令。

第二部分：通过书写一个UserAPP(history.b)文件并写入磁盘中，使得每次调用history.b时，能够将文件（.history）的内容全部输出。

第三部分：键入上下键时，切换历史命令（与linux的上下键行为一致）。

注意：

- 禁止使用局部变量或全局变量的形式实现保存历史指令。（这意味着不能用堆栈区实现）
- 禁止在烧录fs.img时烧录一个.history文件。这意味着你需要在第一次写入时，创建一个.history文件，并在随后每次输入时在.history文件末尾写入。

需要关注的核心文件：

- sh.c：需要在命令执行前后把line写进文件
- serv.c：注意serve_open的逻辑（目前的openfile仅支持打开文件，完成easy部分内容后，加上了对O_MKDIR的支持，现在需要加上对O_CREATE的支持。）
- history.c：一个简单的UserApp
- 其他自行设计的数据结构

Challenge部分——exec

任务背景

在lab6的实验中，我们使用了spawn函数实现了生成为新进程（将目标程序加载为新的进程），帮助我们实现了shell。在spawn中，我们将目标程序的相关信息加载给新进程。

任务目标

实现exec函数组，将目标程序加载到“本进程”。

任务要求

查找资料，理解并实现exec系列函数。

> 提示：exec并没有创建新的进程，它所做的是将原进程的代码段、数据段、堆栈段等用目标程序代替，但进程的envid并没有被替换。

需要关注的核心文件：spawn.c

需要增加系统调用（不能“左脚踩右脚”，借助内核态才能帮助我们将代码段替换）。

Challenge部分——实现shell环境变量

任务目标：

实现shell变量，支持 `export [-xr]` 导出环境变量和设置只读变量，支持 `unset` 和 `set` 命令。（建议用内建指令实现这三条指令的简易版本）

支持并在执行诸如 `echo.b $variable` 指令时能显示正确的值。

本部分开放性很强，可以参考linux系统的环境变量。

在Lab6挑战性任务中，Easy部分和Normal部分为必做，还需要从两个Challenge部分中任选一个完成。实现上述任务后，请自行设计展示（需要在申优答辩时展示效果并陈述实现流程）。