你的任务是**独立**实现一个基本的 Unix 命令行 Shell。你的 Shell **必须**使用提供的 C（src-c/ 目录）或 C++ 解析器（src/ 目录），并实现 ISH 手册页中描述的一部分功能。需要注意的是，你**不需要**实现作业控制或管道（pipelines）。

你需要实现的基本功能包括：

基本命令执行

- Shell 提示符应正确显示，符合手册页描述
- 能够运行完整命令名称的命令
- 能够运行带有参数的完整命令名称的命令
- **cd** 内建命令可以正常工作（可用 /bin/pwd 进行检查）
- **quit** 内建命令和 EOF（文件结束符）能正确退出 Shell 并清理资源

环境变量和别名管理

- **环境变量**能够正确传递给子进程
- **PATH 变量**搜索功能正常
- **错误的 PATH 语法**能够被正确处理
- **printenv**（如果 setenv 无参数时）能够正确显示环境变量
- **unsetenv** 能够正确移除环境变量
- 能够正确添加别名
- 能够正确移除别名

其他功能

- **.ishrc** 文件能够正确执行

文件重定向

- 输出重定向到**一个普通文件**能够正常工作
- **追加模式**（appending to a file）能够正常工作
- **标准输出（stdout）和标准错误（stderr）** 的重定向能够正常工作
- **从文件中读取输入**能够正常工作
- **检测到模棱两可的重定向（ambiguous redirections）时应报错**
- **输入和输出同时重定向**能够正常工作

错误处理

- Shell 应该能够**优雅地处理错误**，并在权限或其他问题导致命令执行失败时，像 csh 那样正确报错

你不需要实现的功能（ISH 手册页中提到但不要求实现）

- **作业控制**（& 分隔符、^Z 信号处理，以及 bg / fg / jobs 内建命令）
- **管道（pipelines）**

起始代码和编程限制

你的 Shell **必须使用 C 或 C++ 编写**，并使用 cmake 进行编译，生成可执行文件 ish。

提供的 C++ 解析器（src/ 目录）能够解析 ish 需要实现的功能，但**不解析管道（因为本作业不要求实现管道）**。此外，还提供了 **C 版本的起始代码**（src-c/ 目录），你可以选择使用。如果要切换到 C 版本，你需要修改 CMakeLists.txt，让它指向 src-c/ 目录下的源代码。

你的代码必须满足以下要求：

- **干净整洁** —— 代码应当**无警告（warnings）**，并且在一个标准 UNIX 系统上能够正确编译（需要安装 Boost C++ 库）。
- **只能使用提供的解析器** —— 你**必须**使用提供的 C 或 C++ 解析器来实现 Shell。你可以**修改解析器**（例如添加对内建命令的检测、扩展命令结构等），但整体解析逻辑必须保持一致。你**不能**使用自己编写的解析器，也不能从其他地方获取解析器。
- **必须使用 fork() 和 execve()** 来启动新进程
- **必须使用 dup() 或 dup2()** 来处理文件重定向
- **禁止使用 system()** 或其他 exec 系列的变体，也不能使用它们的 C++ 版本，**必须直接使用 execve()**

测试

- 约 **85% 的最终成绩** 来自于官方提供的**自动评分测试用例**（将在作业发布约一周后提供）

- 其余 **15% 的成绩** 来自于**错误处理测试**（例如检测错误的命令、权限问题等）
- 当测试用例发布时，系统会提交一个 **pull request** 到你的 GitHub 仓库，以添加这些测试

作业提交

- 你需要使用 **GitHub Classroom** 来提交你的代码
- 在**截止日期前**，请确保**所有的修改都已提交并推送（commit & push）**到你的 GitHub 仓库的 main 分支

支持和参考资料

在开始之前，你应该先熟悉 UNIX 手册第 **2 章**（系统调用部分）以及 **第 3 章**（提供了一些便捷的库函数）。然而，你**不能**使用 system() 以及 ISH 手册页中禁止使用的其他函数。

此外，Richard Stevens 的书籍 **《Advanced Programming in the UNIX Environment》**（《UNIX 环境高级编程》）对本作业非常有帮助，**第 7、8、9 章尤其相关**。这本书可以通过 **UNM 图书馆**免费获取。

额外建议

- **小心进程管理**：你的 Shell 需要创建子进程，而子进程可能会继续创建更多的进程。如果你不小心创建了**太多进程**，可能会导致 Unix 资源耗尽，影响整个系统的运行。请务必**谨慎管理进程**，避免导致系统崩溃。
- **尽早开始**：这个作业涉及很多内容，理解清楚要求并提前规划是非常重要的。
- **设计实现计划**：在实现 Shell 之前，建议**先写详细的设计、实现和测试计划**，再开始编码。
- **逐步实现功能**：建议先实现最基础的功能，例如**解析和执行简单命令**，然后再依次添加**环境变量处理**、**PATH 解析**，最后实现**文件 I/O 重定向**等功能。
- **如果不确定行为**：可以参考 csh 的行为来设计你的 Shell。
- **如有问题**：可以在 **Discord** 上提问。

总结

本次作业的目标是让你用 C/C++ **实现一个基本的 Unix Shell**，重点在于**进程管理、环境变量、文件重定向和错误处理**。你需要使用 fork() 和 execve() 来启动进程，使用 dup() 或 dup2() 来管理 I/O 重定向，并使用提供的解析器完成命令解析。作业**不涉及作业控制和管道**。

你需要在 **GitHub 上提交代码**，并在**作业截止前 push 到主分支**。此外，官方测试用例将在作业发布一周后提供。