一. 安装 ROOT 环境并编写程序

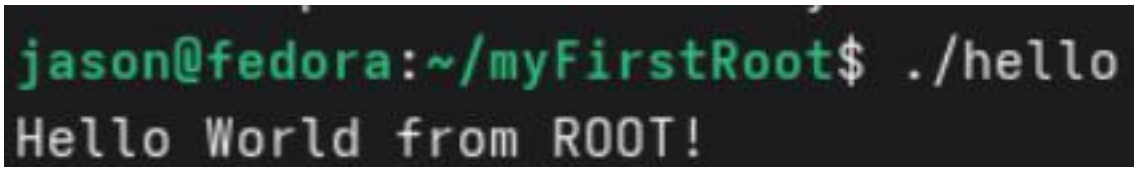

```
jason@fedora:~$ root-config --version
6.38.04
```

-- ROOT 版本号为 6.38.04

```cpp
#include <iostream>
#include "TString.h"

int main() {
    TString message = "Hello World from ROOT!";
    std::cout << message << std::endl;
    return 0;
}
```

--cxx 文件代



编译后成功输出“Hello World from ROOT!“

二. ROOT 存储格式优势分析

ROOT 存储格式的核心优势在于，它专为应对大规模科学数据分析的独特挑战而设计，提供了一个远超普通文本或原始二进制格式的系统化解决方案。其核心优势可概括为以下几点：

1. 自我描述与长期可读：文件采用包含“字典”信息的二进制格式，不仅存储数据，还自动存储描述数据结构的元数据。这使得文件本身具备独立性，确保即使在没有原始代码的情况下，数据也能被长期、正确地解读。

2. 高效存储与快速访问：首先，通过支持先进的压缩算法，在保持高性能的同时显著减少了存储空间占用。其次，其核心的 TTree 容器能将海量结构相同的“事件”数据组织成柱状结构，允许用户高效地随机访问和选择性读取特定数据分支，而无需加载整个文件，极大提升了分析效率。

3. 无缝集成与分析就绪：该格式深度集成于 ROOT 框架的整个分析生态系统。数据从存储、处理到统计分析和可视化，可以在一个统一的框架内完成无缝流转，避免了不同工具和格式间转换的复杂性与风险。

4. 版本控制与向后兼容：支持数据结构的版本管理，确保了软件持续演进过程中，新版本依然能够读取旧版本创建的数据文件，保障了科研数据的长期价值和投资。

三. 拓展二

1. Shell 语言的优势：

1. 实现自动化与批量化：您无需手动为 M 个输入文件逐个创建或修改提交描述文件。通过 Shell 的循环结构（如 for 循环），可以自动遍历指定目录下的所有目标文件，并动态生成相应的作业提交命令。

2. 高效处理任务分发与资源分配：结合命令行参数、变量和条件判断，您可以编写脚本逻辑，根据得到的 CPU 核数 N，智能地将 M 个作业分组或并发提交，以优化资源利用率和任务执行效率。

3. 集成复杂的工作流：整个流程——从文件列表生成、参数替换、到最终调用 condor_submit 命令——可以封装在一个脚本中，一键执行，避免了人工操作可能带来的遗漏和错误。

2. 代码理解：

————基础配置————

universe = vanilla: 指定作业的执行环境为“vanilla”

executable: 指定要在远程计算节点上运行的可执行脚本或程序，此处是一个 Shell 脚本的绝对路径

arguments: 定义传递给上述可执行文件的命令行参数。参数中包含了变量（如 $Fnx(indata)`, `$Fn(indata)），这些变量会在每个具体作业实例化时被替换为对应的输入文件名（不含或包含扩展名），用于动态生成输出文件路径

————资源配置————

RequestCpus: 指定作业请求的 CPU 核心数，为空可能默认为 1

request_disk: 请求 2GB 的磁盘空间

+AccountingGroup: 指定作业所属的记账组，用于资源使用统计和配额管理

————作业标识————

ID = $(ClusterId).$(ProcId): 定义一个变量 ID，它由 HTCondor 系统自动分配的作业集群 ID 和进程 ID 组合而成，确保每个作业实例都有唯一标识符

output, error, log: 分别定义标准输出、标准错误和 HTCondor 自身日志文件的存放路径和命名格式

transfer_input_files: 指定需要从提交节点传输到执行节点的输入文件列表，此处为变量

————具体执行————

通过 queue ... matching files 指令，自动发现指定目录下的所有 .in 文件，并为每一个文件生成并提交一个独立的计算作业。每个作业使用相同的可执行脚本和处理逻辑，但处理不同的输入数据（$(indata)），并生成具有唯一标识的输出文件。