## 实验题目

编写一个C/C++程序，列出用户模式下的所有进程信息

## 实验背景

在Linux操作系统中，进程管理是一个非常重要的任务。ps命令是一个常用的命令行工具，它可以列出当前系统中正在运行的进程的信息。ps -ef命令是一种常见的格式，它以详细的方式显示进程信息。为了实现这一功能，我们可以使用/proc文件系统，其中的status和stat文件包含了有关每个进程的信息。

## 实验原理

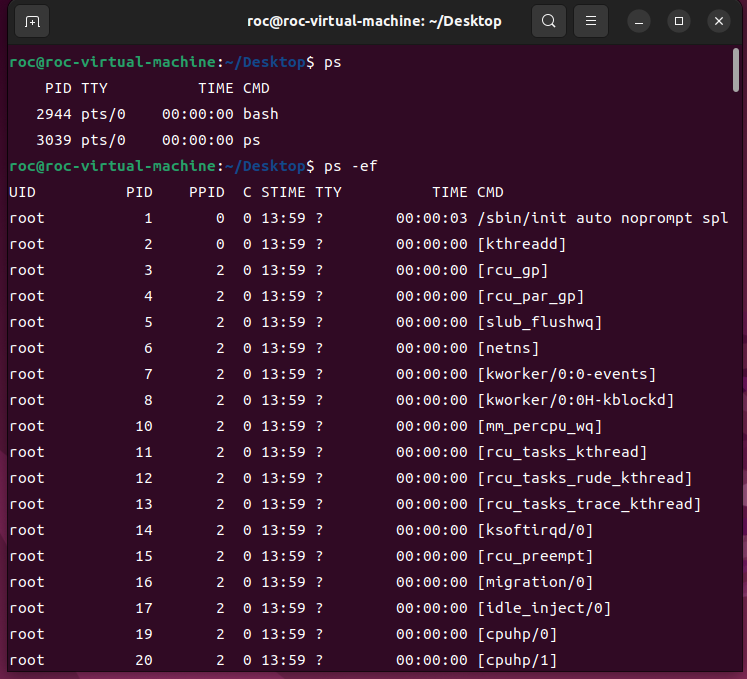
本实验的主要原理是通过编写C/C++程序，利用Linux的/proc文件系统，读取和解析每个进程的status文件，以获取进程的关键信息，如进程ID（PID）、名称、状态和父进程ID（PPID）。

/proc文件系统：Linux操作系统中的虚拟文件系统，包含有关进程和系统状态的信息。

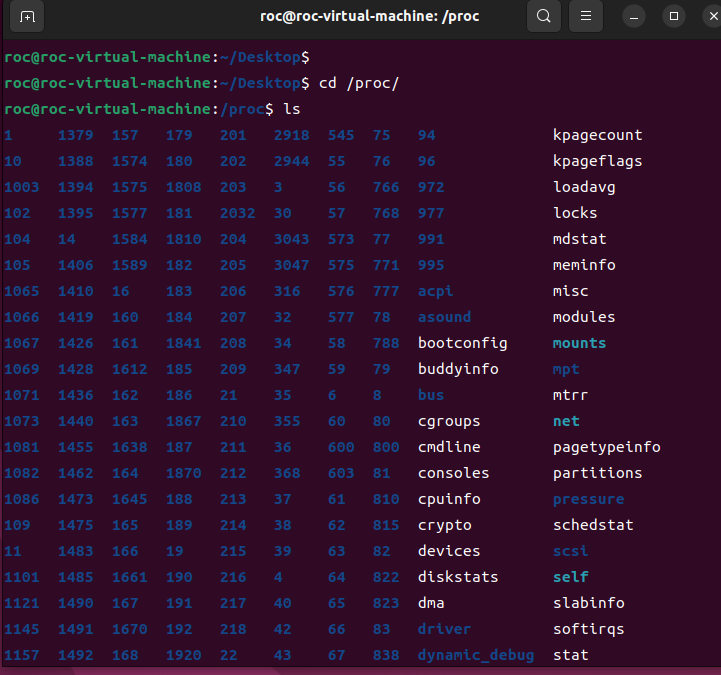
status文件：位于/proc/<pid>/status，包含了进程的详细信息，如名称、状态、PPID等。

stat文件：位于/proc/<pid>/stat，包含了进程的一些状态信息，如进程ID和状态。

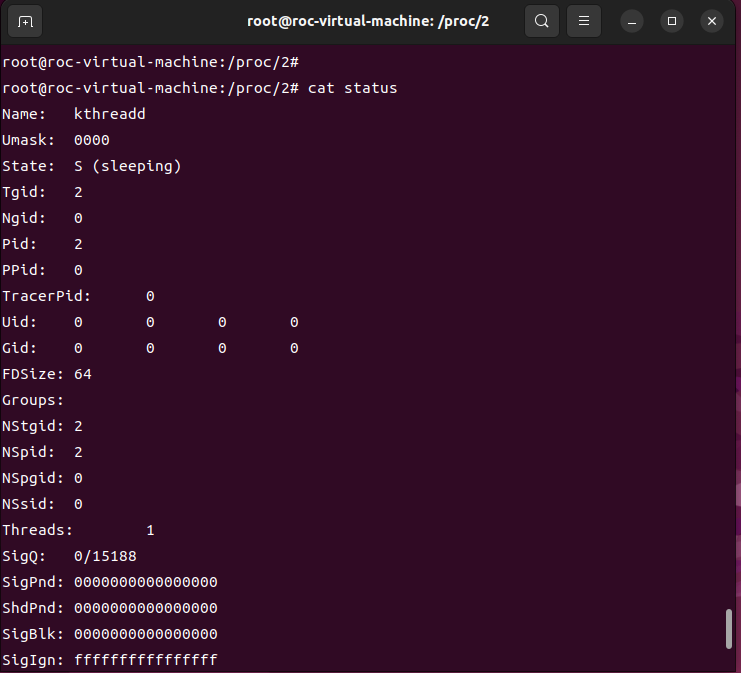
Ps Ps -ef



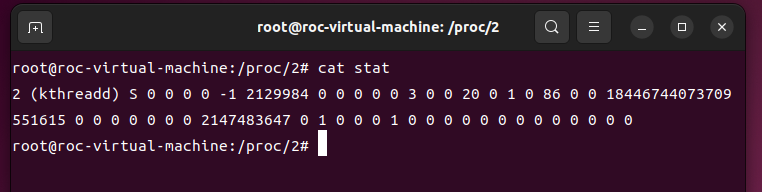
/proc



status文件记录进程信息



Stat文件记录进程信息



## 实验过程

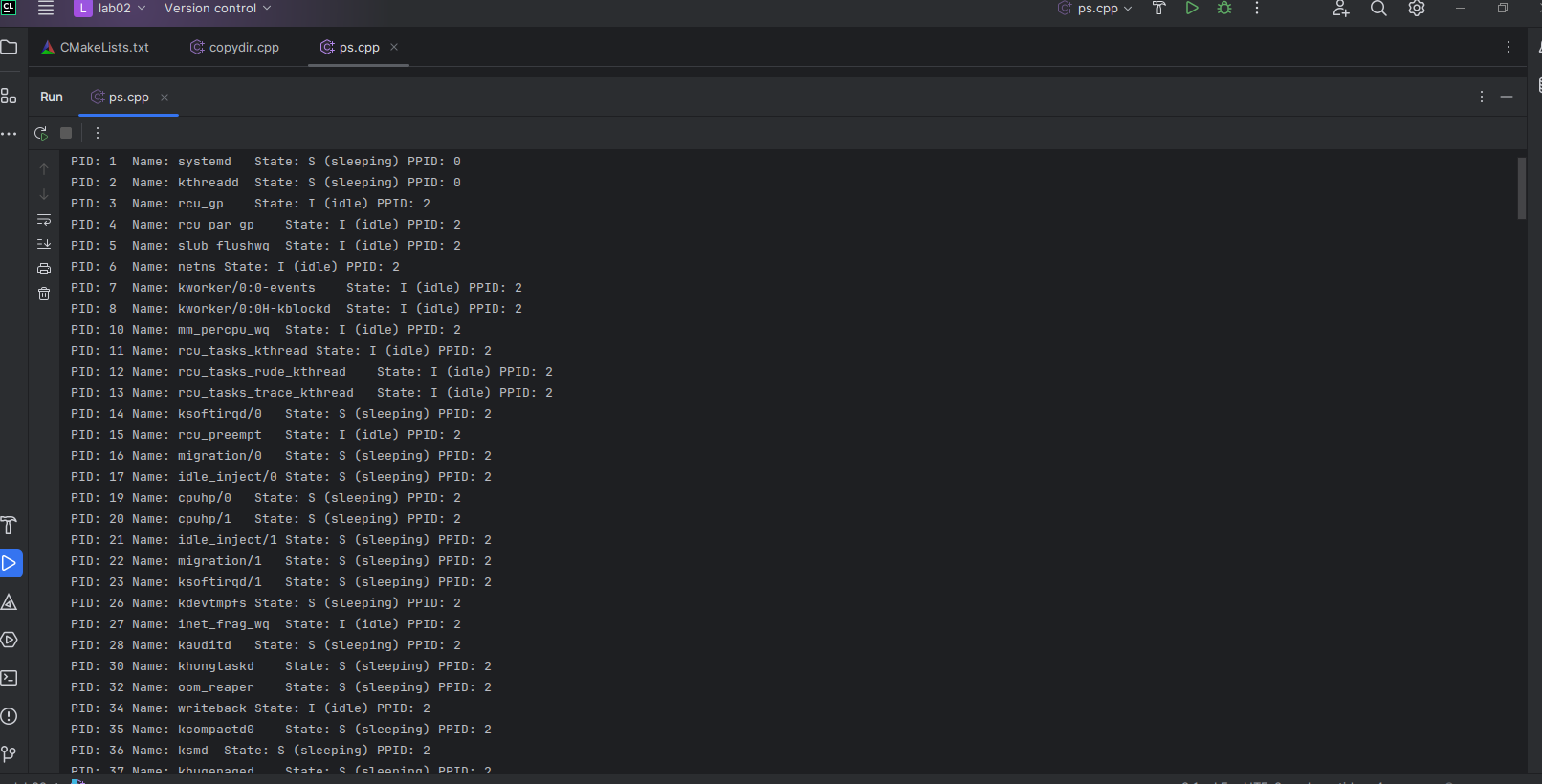
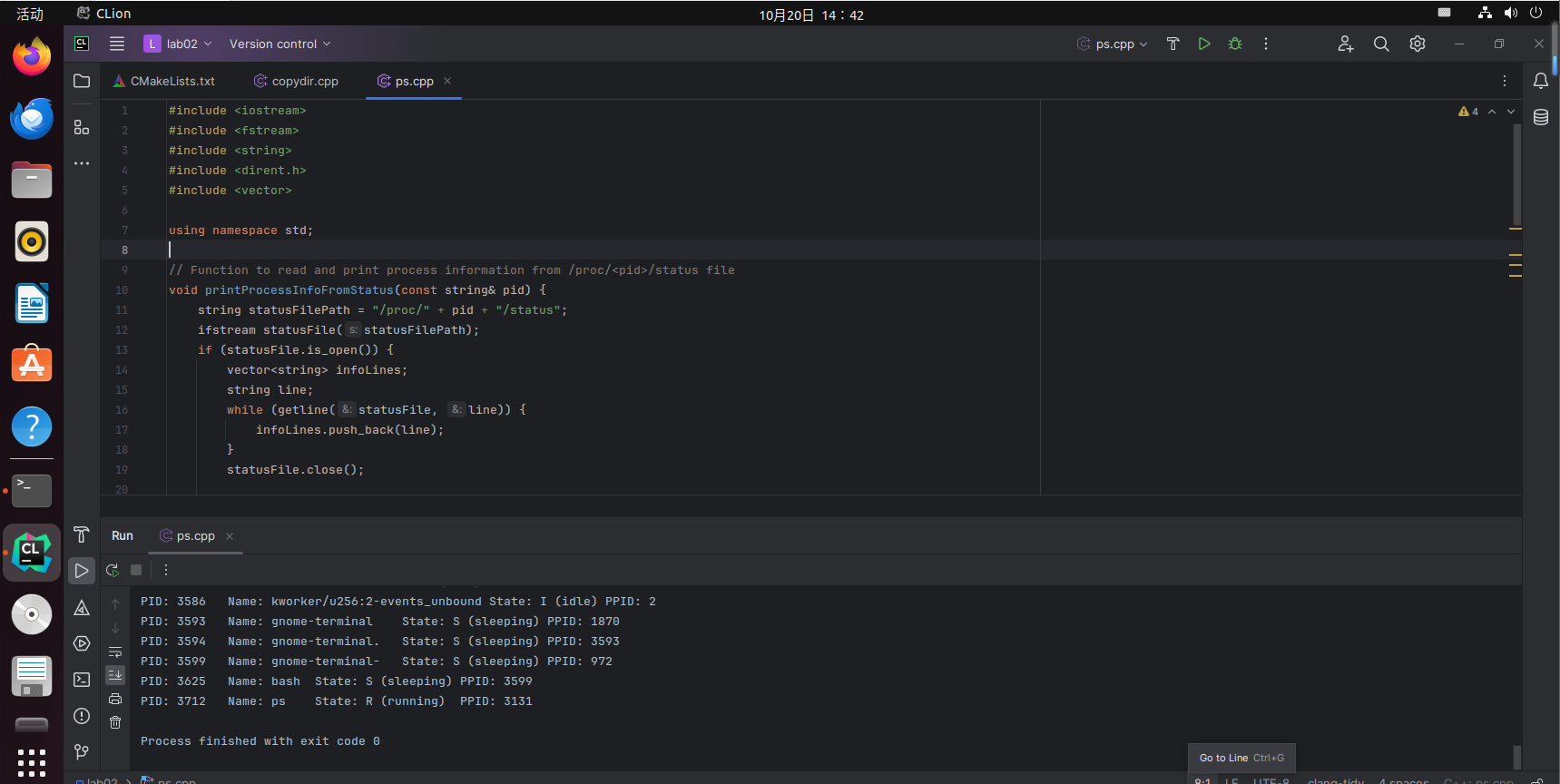
读取status文件并提取信息：编写一个函数printProcessInfoFromStatus，它能够读取/proc/<pid>/status文件，并提取有关进程的信息，如进程名称、状态和父进程ID。这些信息将被存储在一个向量中，然后被格式化并打印出来。

列出所有运行中的进程：编写一个函数listRunningProcesses，用于遍历/proc目录并列出所有运行中的进程。它会检查每个目录项，筛选出目录名为数字（表示进程ID）的目录，并调用printProcessInfoFromStatus函数来打印有关该进程的信息。

主函数：在主函数中，我们调用listRunningProcesses函数以列出所有运行中的进程。输出将包括每个进程的PID、名称、状态和父进程ID。

## 实验结果

编译并运行程序后，它将输出所有用户模式下运行的进程的信息，包括PID、名称、状态和父进程ID。这些信息以表格形式呈现，类似于ps -ef命令的输出。



PID Name State PPID

-------------------------------------------

PID: 1 Name: init State: S PPID: 0

PID: 2 Name: kthreadd State: I PPID: 0

PID: 3 Name: rcu\_gp State: I PPID: 2

...

## 实验总结

通过本次实验，我们学习了如何使用C/C++编程语言编写程序，模拟了类似于ps和ps -ef命令的功能，以列出用户模式下的所有进程。我们还深入了解了Linux操作系统中的/proc文件系统，以及如何通过读取status文件来获取有关进程的详细信息。

此外，实验中我们学到了如何使用文件操作函数和目录操作函数来访问系统文件和目录。这些技能对于系统编程和操作系统课程非常有用。

通过这个实验，提高了对Linux进程管理和系统编程的理解，为进一步学习和应用操作系统相关的知识打下了坚实的基础。

## 附录：

源代码

#include <iostream>

#include <fstream>

#include <string>

#include <dirent.h>

#include <vector>

using namespace std;

// Function to read and print process information from /proc/<pid>/status file

void printProcessInfoFromStatus(const string& pid) {

string statusFilePath = "/proc/" + pid + "/status";

ifstream statusFile(statusFilePath);

if (statusFile.is\_open()) {

vector<string> infoLines;

string line;

while (getline(statusFile, line)) {

infoLines.push\_back(line);

}

statusFile.close();

if (infoLines.size() > 0) {

// Extract and format the relevant information (e.g., Name, State, PID)

string name, state, ppid;

for (const string& infoLine : infoLines) {

if (infoLine.find("Name:") != string::npos) {

name = infoLine.substr(infoLine.find(":") + 2);

} else if (infoLine.find("State:") != string::npos) {

state = infoLine.substr(infoLine.find(":") + 2);

} else if (infoLine.find("PPid:") != string::npos) {

ppid = infoLine.substr(infoLine.find(":") + 2);

}

}

cout << "PID: " << pid << "\tName: " << name << "\tState: " << state << "\tPPID: " << ppid << endl;

}

} else {

cout << "Failed to open " << statusFilePath << endl;

}

}

// Function to list all running processes

void listRunningProcesses() {

DIR\* procDir = opendir("/proc");

if (procDir == nullptr) {

cout << "Failed to open /proc directory." << endl;

return;

}

dirent\* entry;

while ((entry = readdir(procDir)) != nullptr) {

// Check if the entry is a directory and its name is a number (i.e., a process ID)

if (entry->d\_type == DT\_DIR && isdigit(entry->d\_name[0])) {

printProcessInfoFromStatus(entry->d\_name);

}

}

closedir(procDir);

}

int main() {

std::cout << "Listing running processes:\n" << std::endl;

listRunningProcesses();

return 0;

}