**操作系统程序作业报告**

姓名:禹泽海 李博昊 学号:2021302111 2021302009

**一、程序要求及信息**

探究操作系统的可实现进程并发最大数

程序语言：c语言

程序运行环境：ubuntu 22.04.1

运行用户：分别在wsl和VMware运行做出比较

程序代码：

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

int max\_processes = 0;

while (1) {

pid\_t pid = fork();

if (pid < 0) {

// Error forking

break;

} else if (pid == 0) {

// Child process

exit(0);

} else {

// Parent process

max\_processes++;

}

}

printf("Maximum concurrent processes: %d\n", max\_processes);

return 0;

}

代码中需要解释的地方：

1 .pid\_t ：Process ID

它是在头文件 <sys/types.h> 中定义的，用于表示进程标识符。

在C语言中，pid\_t 是一个整数类型，通常是一个有符号整数。具体的大小和符号性质取决于系统的实现。在大多数系统上，pid\_t 是一个32位或64位整数。

2 .fork函数：

调用 fork() 函数时，操作系统会创建一个新的进程，称为子进程，该子进程是调用进程的副本，包括代码、数据和堆栈。父进程是调用 fork() 函数的进程。

fork() 函数会返回两次。在父进程中，fork() 返回新创建的子进程的PID（Process ID），表示子进程的唯一标识符；在子进程中，fork() 返回0，表示当前是子进程执行代码。

这样，通过判断 fork() 的返回值，可以在父进程和子进程中执行不同的代码。

3 .如何实现进程并发：

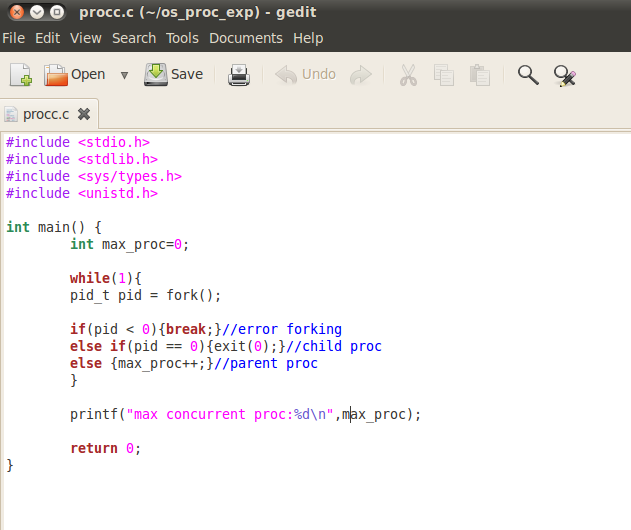
通过不断调用 fork() 函数，在父进程和子进程之间创建新的进程。子进程执行 exit(0) 退出，而父进程则继续循环创建新进程并统计进程数。最后输出最大同时存在的进程数。每次调用 fork() 都会将当前进程复制一份，形成一个新的进程，这样就实现了进程共存。

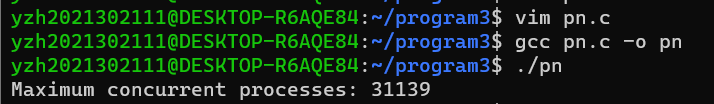
讲解可能有些抽象，关于fork的细节我再linux编程学习网站上进行了解：https://xiaoxiami.gitbook.io/linux-server/duo-jin-cheng-bian-cheng/forkhan-shu

**二、代码程序运行全过程：**

**Step1**

Vim pn.c文件并写入代码，进行编译并运行





可以得到当前实验结论：31139

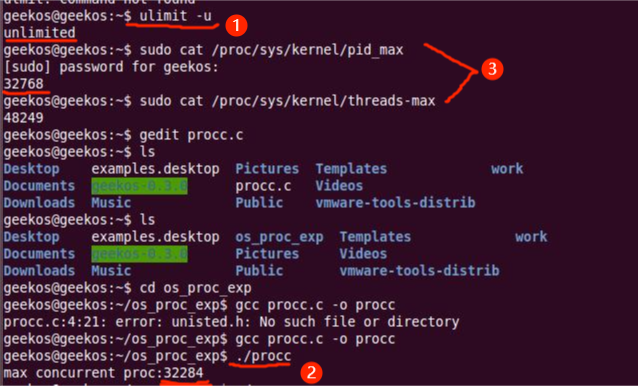
除此之外，我还发现了其他方法去查询相关进程数

1 .Ulimit -u指令：查询当前用户可创建的最大进程数量，代表用户级，如果结果为ulimited，则意味着只要不超过系统级最大线程数即可。

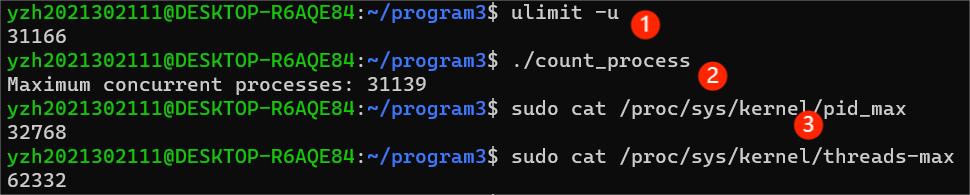
2 .sudo cat指令：这条指令查询系统中的进程最大数，是系统级的参数查询

以下分别介绍在VMware geekos和wsl下运行结果对比：

VM：



WSL：



sudo cat /proc/sys/kernel/pid\_max 查询结果系统级进程最大数

sudo cat /proc/sys/kernel/threads-max查询结果系统级最大线程数

从中发现两台虚拟机虽然实验结果各异，但他们的系统级最大进程数都是