**实验13**

1. **多线程版本实现**

电脑的屏幕截图

AI 生成的内容可能不正确。

源代码

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int is\_prime(long long n) {

    if (n <= 1) return 0;

    if (n <= 3) return 1;

    if (n % 2 == 0 || n % 3 == 0) return 0;

    for (long long i = 5; i \* i <= n; i += 6) {

        if (n % i == 0 || n % (i + 2) == 0) return 0;

    }

    return 1;

}

typedef struct {

    long long start;

    long long result;

} thread\_data;

void\* find\_prime(void\* arg) {

    thread\_data\* data = (thread\_data\*)arg;

    for (long long i = data->start; ; i++) {

        if (is\_prime(i)) {

            data->result = i;

            pthread\_exit(NULL);

        }

    }

    return NULL;

}

int main() {

    const int num\_threads = 5;

    pthread\_t threads[num\_threads];

    thread\_data data[num\_threads];

    // 创建线程

    for (int i = 0; i < num\_threads; i++) {

        data[i].start = (i + 1) \* 100000000LL;

        if (pthread\_create(&threads[i], NULL, find\_prime, &data[i]) != 0) {

            perror("pthread\_create");

            exit(EXIT\_FAILURE);

        }

    }

    // 等待线程完成

    for (int i = 0; i < num\_threads; i++) {

        pthread\_join(threads[i], NULL);

        printf("Thread %d found prime: %lld\n", i+1, data[i].result);

    }

    return 0;

}

1. **多进程版本实现**

文本

AI 生成的内容可能不正确。

源代码

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

int is\_prime(long long n) {

    if (n <= 1) return 0;

    if (n <= 3) return 1;

    if (n % 2 == 0 || n % 3 == 0) return 0;

    for (long long i = 5; i \* i <= n; i += 6) {

        if (n % i == 0 || n % (i + 2) == 0) return 0;

    }

    return 1;

}

void find\_prime(long long start) {

    for (long long i = start; ; i++) {

        if (is\_prime(i)) {

            printf("%lld\n", i);

            exit(0);

        }

    }

}

int main() {

    const int num\_processes = 5;

    pid\_t pids[num\_processes];

    int pipes[num\_processes][2];

    // 创建管道

    for (int i = 0; i < num\_processes; i++) {

        if (pipe(pipes[i]) == -1) {

            perror("pipe");

            exit(EXIT\_FAILURE);

        }

    }

    // 创建子进程

    for (int i = 0; i < num\_processes; i++) {

        pids[i] = fork();

        if (pids[i] == -1) {

            perror("fork");

            exit(EXIT\_FAILURE);

        }

        if (pids[i] == 0) { // 子进程

            close(pipes[i][0]); // 关闭读端

            dup2(pipes[i][1], STDOUT\_FILENO); // 重定向输出到管道

            find\_prime((i + 1) \* 100000000LL);

            exit(EXIT\_SUCCESS);

        } else { // 父进程

            close(pipes[i][1]); // 关闭写端

        }

    }

    // 父进程读取结果

    for (int i = 0; i < num\_processes; i++) {

        char buffer[20];

        read(pipes[i][0], buffer, sizeof(buffer));

        printf("Process %d found prime: %s", i+1, buffer);

        close(pipes[i][0]);

    }

    // 等待所有子进程结束

    for (int i = 0; i < num\_processes; i++) {

        waitpid(pids[i], NULL, 0);

    }

    return 0;

}

1. **编译程序**

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

1. **测量运行时间**

多进程

文本

AI 生成的内容可能不正确。

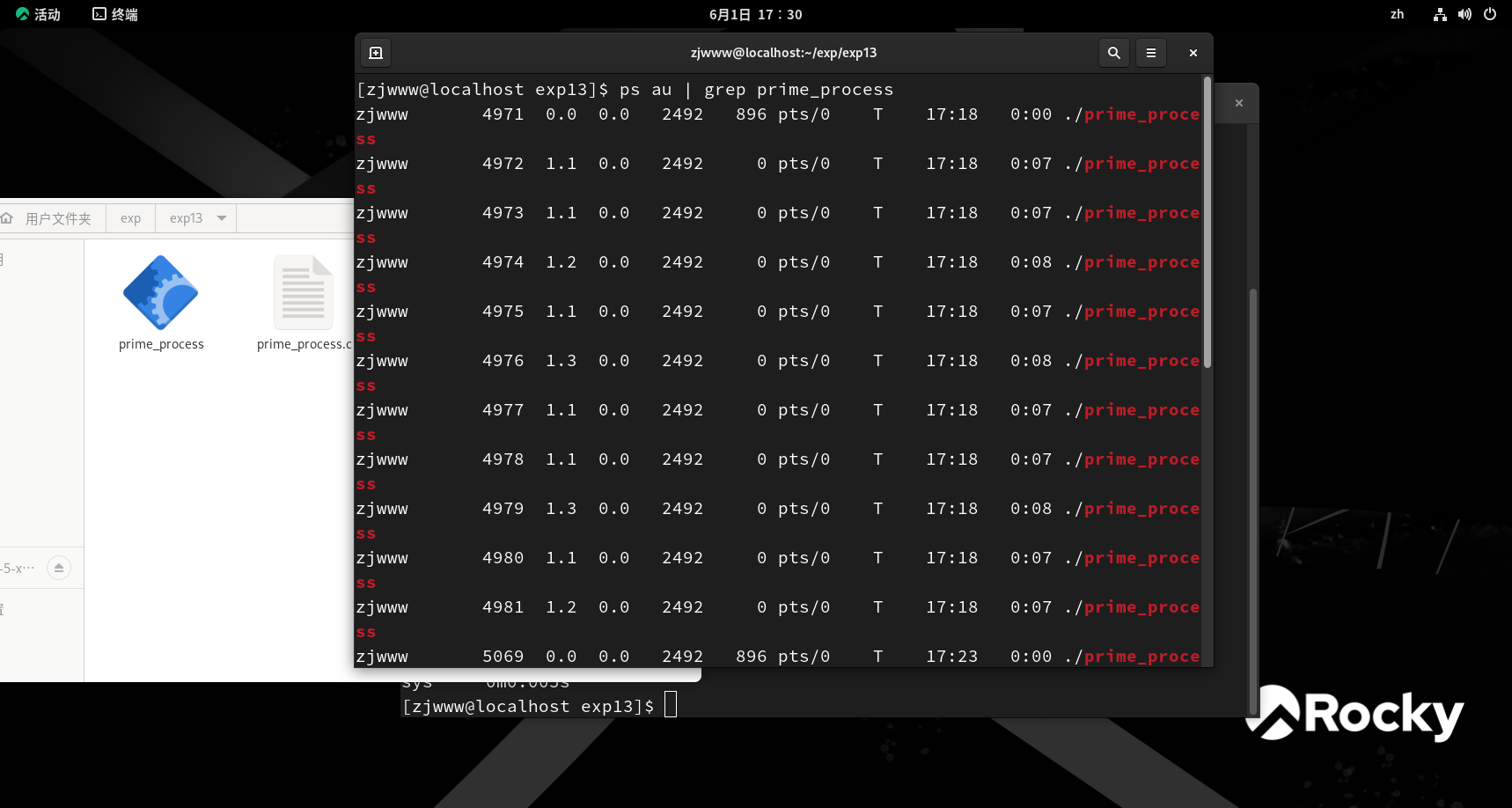
多线程

文本

AI 生成的内容可能不正确。

1. **测量内存占用**

多进程



图形用户界面

AI 生成的内容可能不正确。

多线程

图形用户界面

AI 生成的内容可能不正确。