**实验11**

1. **源代码**

**图形用户界面

AI 生成的内容可能不正确。**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>  // 添加wait函数的头文件

#include <signal.h>

#include <time.h>

int sum = 0; // 全局变量，用于累加

int pipefd[2]; // 将管道文件描述符设为全局变量

// 信号处理函数

void sigusr1\_handler(int signo) {

    int num;

    read(pipefd[0], &num, sizeof(int)); // 从管道读取数据

    sum += num;

}

int main() {

    pid\_t pid;

    // 创建匿名管道

    if (pipe(pipefd) == -1) {

        perror("pipe");

        exit(EXIT\_FAILURE);

    }

    // 设置SIGUSR1信号处理函数

    signal(SIGUSR1, sigusr1\_handler);

    // 创建子进程

    pid = fork();

    if (pid == -1) {

        perror("fork");

        exit(EXIT\_FAILURE);

    }

    if (pid == 0) { // 子进程

        close(pipefd[0]); // 关闭读端

        srand(time(NULL)); // 初始化随机数种子

        while (1) {

            int num = rand() % 9 + 1; // 生成1~9的随机数

            write(pipefd[1], &num, sizeof(int)); // 写入管道

            kill(getppid(), SIGUSR1); // 向父进程发送信号

            sleep(2); // 等待2秒

        }

        close(pipefd[1]); // 关闭写端

        exit(EXIT\_SUCCESS);

    } else { // 父进程

        close(pipefd[1]); // 关闭写端

        while (1) {

            printf("Current sum: %d\n", sum);

            sleep(1); // 等待1秒

        }

        close(pipefd[0]); // 关闭读端

        wait(NULL); // 等待子进程结束

        exit(EXIT\_SUCCESS);

    }

    return 0;

}

1. **编译和运行**

**文本

AI 生成的内容可能不正确。图形用户界面, 文本

AI 生成的内容可能不正确。**