**中国计量大学**

**《计算机操作系统》**

**实验报告书**

**（实验三进程通信）**

**班 级 22计算机3**

**学 号 2200303310**

**姓 名 陈忠鹏**

**日 期 2024/11/6**

成绩： 教师签字：

1. **实验目的和意义**

（正文，小五，宋体，单倍行间距）

（正文，小五，宋体，单倍行间距）

以下正文都按这个格式。

1. **实验方案设计及实验过程**

**2.1 实验背景及内容**

**2.2 实验设计及方案**

**2.2.1 数据结构设计（可选）**

**猜拳结构体设计**

**struct Game {**

**int Round;**

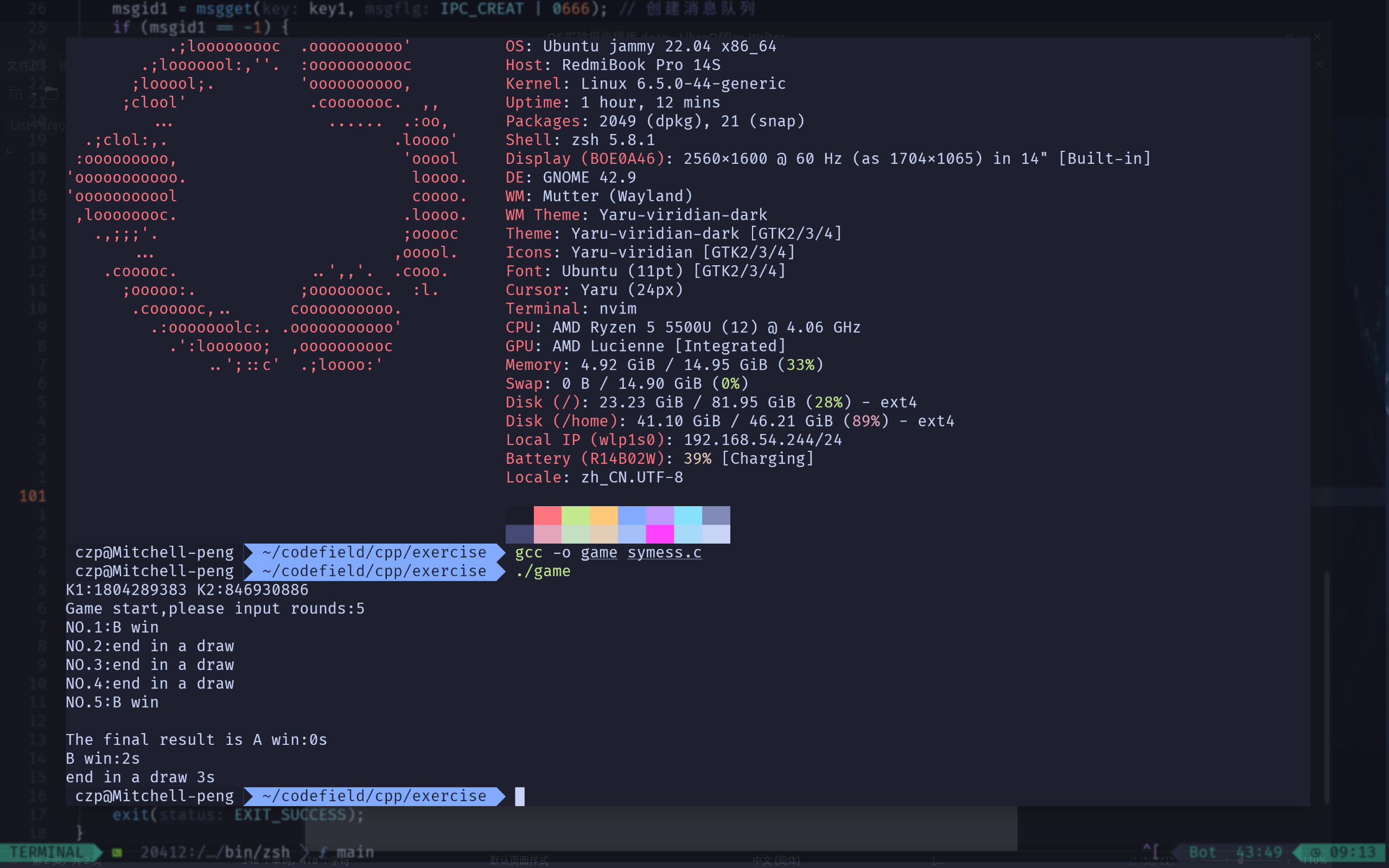
**long Type;**

**};**

* + 1. **算法详述（或设计方案）**

使用消息队列的形式传输信息，让主进程获取消息队列的消息并进行结果判定。

1. **程序测试结果与分析**

****

1. **实验总结**
2. **参考文献**

[1]汤小丹等,《计算机操作系统》(摹课版)(M),北京：人民邮电出版社，2021年

[2]王红玲，褚晓敏,《计算机操作系统实验指导》(M),北京：人民邮电出版社，2021年

1. **附录（源程序，可选）（六号Time New Roman字体，单倍行间距）**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/ipc.h>

#include <sys/msg.h>

struct Game {

int Round;

long Type;

};

void result\_send(int num) {

struct Game game;

game.Type = 1;

game.Round = rand() % 3;

msgsnd(num, &game, sizeof(int), 0);

}

int result\_announce(int a, int b) {

if ((a + 1 == b) || (a - 3 == b))

return -1;

else if (a == b)

return 0;

else

return 1;

}

void writeFile(int \*result\_list, int len) {

int count\_A = 0;

int count\_B = 0;

int pingju = 0;

FILE \*fin;

if ((fin = fopen("result.txt", "w")) == NULL)

printf("This file wasn't opened");

int i;

for (i = 0; i < len; i++) {

switch (result\_list[i]) {

case -1: {

count\_A++;

fprintf(fin, "NO.%d:A win\n", i + 1);

printf("NO.%d:A win\n", i + 1);

break;

}

case 0: {

pingju++;

fprintf(fin, "NO.%d:end in a draw\n", i + 1);

printf("NO.%d:end in a draw\n", i + 1);

break;

}

case 1: {

count\_B++;

fprintf(fin, "NO.%d:B win\n", i + 1);

printf("NO.%d:B win\n", i + 1);

break;

}

}

}

printf("\nThe final result is A win:%ds \nB win:%ds \nend in a draw %ds\n",

count\_A, count\_B, pingju);

fprintf(fin,

"\nThe final result is A win:%ds \nB win:%ds \nend in a draw %ds\n",

count\_A, count\_B, pingju);

fclose(fin);

}

int main() {

int times;

srand((time\_t)NULL);

int key1 = rand();

int key2 = rand();

printf("K1:%d K2:%d\n", key1, key2);

int \*result\_list;

pid\_t pid1, pid2;

int msgid1, msgid2;

msgid1 = msgget(key1, IPC\_CREAT | 0666); // 创建消息队列

if (msgid1 == -1) {

fprintf(stderr, "failed with error");

exit(EXIT\_FAILURE);

}

msgid2 = msgget(key2, IPC\_CREAT | 0666); // 创建消息队列

if (msgid2 == -1) {

fprintf(stderr, "failed with error");

exit(EXIT\_FAILURE);

}

printf("Game start,please input rounds:");

scanf("%d", &times);

result\_list = (int \*)malloc(times \* sizeof(int));

int i;

for (i = 0; i < times; i++) {

pid1 = fork();

if (pid1 == 0) {

srand((unsigned)time(NULL) \* 300);

result\_send(msgid1);

exit(-1);

}

pid2 = fork();

if (pid2 == 0) {

srand((unsigned)time(NULL) \* i);

result\_send(msgid2);

exit(-1);

}

if (pid1 < 0 || pid2 < 0) {

fprintf(stderr, "Fork Failed");

exit(-1);

} else {

wait(NULL);

wait(NULL);

struct Game game1;

struct Game game2;

msgrcv(msgid1, &game1, sizeof(game1) - sizeof(long), 0, 0);

msgrcv(msgid2, &game2, sizeof(game2) - sizeof(long), 0, 0);

int j = result\_announce(game1.Round, game2.Round);

result\_list[i] = j;

}

}

writeFile(result\_list, times);

exit(EXIT\_SUCCESS);

}