实验报告（五）

李国楷 22  126

1.实验名称：实验 5 编制模拟“五个哲学家”问题的程序

2.实验内容描述

1、程序语法

philosopher [-t <time>]

<time>是哲学家进餐和沉思的持续时间值，缺省值为 2 秒。

2、五个哲学家的编号为 0~4，分别用五个进程***独立***模拟。

3、程序的输出要简洁，仅输出每个哲学家进餐和沉思的信息。例如，当编号为 3 的哲学家在进餐时，就打印：

philosopher 3 is eating

而当他在沉思时，则打印：

philosopher 3 is thinking

除此之外不要输出其他任何信息。

4、利用课堂已教授的知识而不使用线程或 IPC 机制进行同步。

5、程序应该一直运行，直到人为地终止它（按 Ctrl-C 或 Ctrl-\）。不允许出现僵尸进程。

3.设计原理

main 函数部分：

```
int main(int argc, char * argv[]){
    int i;
    pid_t pid;

    /* 初始化叉子 */
    for(i = 0; i < N; i++) {
        initlock(forks[i]);
    }
```

```
    /* 处理输入 */
    if(argc == 3 && strcmp(argv[1], "-t") == 0) {
        nsecs = atoi(argv[2]);
        // if (!nsecs) err_quit("usage: philosopher [ -t <time> ]");
    } else if (argc != 1) {
        err_quit("usage: philosopher [ -t <time> ]");
    }

    /* 创建五个子进程来模拟五个哲学家 */
    for(i = 0; i < N; i++) {
        pid = fork();
        if (pid == 0) {
            philosopher(i);
        } else if (pid < 0) {
            err_quit("fork error");
        }
    }

    wait(NULL); /* 注意，如果父进程不等待子进程的结束，*/
    /* 那么需要终止程序运行时，就只能从控制台删除在后台运行的哲学家进程 */
}
```

根据指导文件编写，模拟叉子并运用 fork 函数来创建五个进程即可

其他函数：

```
void takeFork(int i) {
    if(i == N-1) { // 人为设定第 N-1 位哲学家是右撇子
    lock(forks[0]);
        lock(forks[i]);
    } else { // 其他是左撇子
    lock(forks[i]);
        lock(forks[i+1]);
    }
}

/* 放下叉子 */
void putFork(int i) {
    if(i == N-1) {
        unlock(forks[0]);
        unlock(forks[i]);
    }
    else {
```

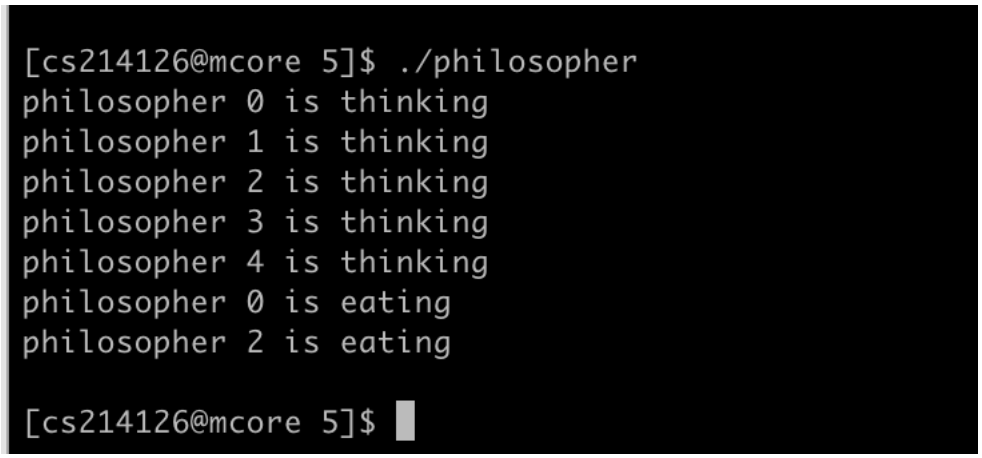
```
        unlock(forks[i]);
        unlock(forks[i+1]);
    }
}

void thinking(int i, int nsecs) {
    printf("philosopher %d is thinking\n", i);
    sleep(nsecs);
}

void eating(int i, int nsecs) {
    printf("philosopher %d is eating\n", i);
    sleep(nsecs);
}

/* 哲学家行为 */
void philosopher(int i) {
    while(1) {
        thinking(i, nsecs);    // 哲学家 i 思考 nsecs 秒
    takeFork(i);        // 哲学家 i 拿起叉子
    eating(i, nsecs);    // 哲学家 i 进餐 nsecs 秒
    putFork(i);        // 哲学家 i 放下叉子
  }
}
```

4.实验结果：

实验文件通过 Makefile 生成

```
philosopher: philosopher.o error.o lock.o
        gcc -g philosopher.o error.o lock.o -o philosopher -L./lib -lapue -lm

philosopher.o: philosopher.c apue.h lock.h
        gcc -g -c philosopher.c

lock.o:lock.c
        gcc -g -c lock.c

error.o: error.c apue.h
        gcc -g -c error.c

clean:
        rm -f philosopher *.o
```

实验结果如下：（按 ctrl + C 人为终止程序）



5.体会和建议

本次实验用简单的程序模拟了五个子程序的运行，教会了我们发现锁死、活锁这些编程中不易发现的 bug 并加以解决，本次实验收获颇丰

6.实验人姓名：李国楷 完成时间 2023.12.10