作业一

作业发布时间：2021/09/19 周日

本次作业要求如下：

1. **截止日期：2021/09/29周三晚24:00**
2. **作业提交方式见群文件**
3. **命名格式**：作业统一命名为“第一次作业+学号+姓名”，**作业为 PDF 格式**；
4. 注意：
   1. 本次作业包括两部分，分别包含**8个题目**和**5个题目**。
   2. 选择只需要写答案，大题要求**有详细过程**，过程算分。
   3. 答案请用另一种颜色的笔回答，便于批改，**否则视为无效答案。**
   4. 大题的过程最好在纸上写了拍照，放到word里。
5. 本次作业遇到问题请联系群里的助教。

Part1 进程与线程

**习题**

1. 进程和程序的一个本质区别是 ( D )

A. 前者分时使用CPU，后者独占CPU B. 前者存储在内存，后者存储在外存

C. 前者在一个文件中，后者在多个文件中 D. 前者为动态的，后者为静态的

1. 下面所列进程的三种基本状态之间的转换不正确的是( B )

A. 就绪状态->执行状态 B. 就绪状态->阻塞状态

C. 执行状态->阻塞状态 D. 执行状态->就绪状态

1. 为什么进程切换的代价要比线程切换要大 ( C )

A. 因为进程切换要切换栈

B. 因为进程切换要切换控制块数据结构

C. 因为进程切换要切换段表

D. 因为进程切换要切换PC指针

1. 下列选项中，不可能在用户态发生的是（ C ）

A. 系统调用 B. 外部中断

C. 进程切换 D. 缺页

1. 在下述父进程和子进程的描述中，正确的是（ A ）。

A. 撤销父进程时，应该同时撤销子进程

B. 父进程和子进程不可以并发执行

C. 撤销子进程时，应该同时撤销父进程

D. 父进程创建了子进程，因而父进程执行完后，子进程才能运行

1. 下列关于线程的叙述中，正确的是（ C ）
2. 采用轮转调度算法时，一进程拥有10个用户级线程，则在系统调度执行时间上占用10个时间片
3. 属于同一个进程的各个线程共享栈空间
4. 同一进程中的线程可以并发执行，但不同进程的线程不可以并发执行
5. 线程的切换，不会引起进程的切换

A. Ⅰ、Ⅱ、Ⅲ B. 仅Ⅱ、Ⅳ C. 全错 D. 仅Ⅱ、Ⅲ

1. 判断以下哪个输出是不正确的？ ( D )



1. L0, Bye, L1, Bye, L2, Bye
2. L0, L1, Bye, L2, Bye, Bye
3. L0, Bye, L1, L2, Bye, Bye
4. L0, Bye, L1, Bye, Bye, L2
5. 分析程序homework\_wait.c，回答下列问题：

|  |  |
| --- | --- |
|  | /\* homework\_wait.c \*/ |
|  | void homework\_wait() { |
|  | pid\_t pid[N]; |
|  | int i, child\_status; |
|  | for (i = 0; i < N; i++) { |
|  | if ((pid[i] = fork()) == 0) { |
|  | exit(100+i); /\* Child \*/ |
|  | } |
|  | } |
|  | printf("hello!\n"); |
|  | for (i = 0; i < N; i++) { /\* Parent \*/ |
|  | pid\_t wpid = wait(&child\_status); |
|  | if (WIFEXITED(child\_status)) |
|  | printf("Child %d terminated with exit status %d\n", |
|  | wpid, WEXITSTATUS(child\_status)); |
|  | else |
|  | printf("Child %d terminate abnormally\n", wpid); |
|  | } |
|  | } |

1. 注释掉第7行代码后，程序执行到第10行，输出多少个“hello!”（用一个N的函数给出答案）? 2^N
2. N=2时，程序正常运行两次，得到的结果是否相同？若不同，请解释原因；

结果不一定相同。因为wait函数等待当前子进程序列的任意一个进程（无序的）终止即返回。

1. 修改程序，使得子进程能够按照其创建的顺序退出。（waitpid.c）

答案1：

|  |  |
| --- | --- |
|  | /\* homework\_wait\_answer1.c \*/ |
|  | void homework\_wait() { |
|  | pid\_t pid[N]; |
|  | int i, child\_status; |
|  | for (i = 0; i < N; i++) { |
|  | if ((pid[i] = fork()) == 0) { |
|  | exit(100+i); /\* Child \*/ |
|  | } |
|  | } |
|  | printf("hello!\n"); |
|  | for (i = 0; i < N; i++) { /\* Parent \*/ |
|  | pid\_t wpid = waitpid(pid[i], &child\_status, 0); |
|  | if (WIFEXITED(child\_status)) |
|  | printf("Child %d terminated with exit status %d\n", |
|  | wpid, WEXITSTATUS(child\_status)); |
|  | else |
|  | printf("Child %d terminate abnormally\n", wpid); |
|  | } |
|  | } |

答案2：

|  |  |
| --- | --- |
|  | /\* homework\_wait\_answer2.c \*/ |
|  | void homework\_wait() { |
|  | pid\_t pid[N], wpid; |
|  | int i, child\_status; |
|  | for (i = 0; i < N; i++) { |
|  | if ((pid[i] = fork()) == 0) { |
|  | exit(100+i); /\* Child \*/ |
|  | } |
|  | } |
|  | printf("hello!\n"); |
|  |  |
|  | /\*回收所有子进程之后再调用waitpid就返回-1，且设置errno为ECHILD \*/ |
|  | i = 0; |
|  | while((wpid = waitpid(pid[i++], &status, 0)) > 0) { |
|  | if (WIFEXITED(child\_status)) |
|  | printf("Child %d terminated with exit status %d\n", |
|  | wpid, WEXITSTATUS(child\_status)); |
|  | else |
|  | printf("Child %d terminate abnormally\n", wpid); |
|  | } |
|  |  |
|  | if (errno != ECHILD) |
|  | unix\_error("waitpid error"); |
|  |  |
|  | exit(0); |
|  | } |

**Part 2. 并发与同步**

1. 一个正在访问临界资源的进程由于申请等待I/O操作而被中断时，它**（ ）**
2. 允许其他进程进入与该进程相关的临界区
3. 不允许其他进程进入任何临界区
4. 允许其他进程抢占处理器，但不得进入该进程的临界区
5. 不允许任何进程抢占处理器
6. 设与某资源相关联的信号量初值为3，当前值为1，若M表示该资源此时可以用的个数，N表示等待该资源的进程数，那么 M 和N 分别为**（ ）**
7. 0，1
8. 1，0
9. 1，2
10. 2，0
11. 如下所示的程序在运行后永远也不会结束。请改写该程序，在不删除和修改任何现有语句的前提下，只添加新语句，使得程序在用户输入任意字符后能够正常结束。并要求程序在结束前只可输出一条语句“received a signal\n”。（提示：使用信号；已给出所需要的头文件；建议实际运行进行测试）

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <sys/types.h> |
|  | #include <signal.h> |
|  | #include <unistd.h> |
|  | #include <stdlib.h> |
|  | #include <sys/wait.h> |
|  | int main(){ |
|  | int child\_pid; |
|  | if((child\_pid=fork())==0){ |
|  | while(1); |
|  | printf("forbidden zone\n"); |
|  | exit(0); |
|  | } |
|  | else{ |
|  | while(getc(stdin)){ |
|  | wait(0); |
|  | exit(0); |
|  | } |
|  | } |
|  | } |

1. 三个进程、、互斥使用一个包含N（N>0）个单元的缓冲区。每次使用produce()生成一个正整数并用put()送入缓冲区某一空单元；每次用get\_odd()从该缓冲区取出一个奇数，然后用count\_odd()统计已经获取的奇数个数；每次用get\_even()从该缓冲区取出一个偶数，然后用count\_even()统计已经获取的偶数个数。请用信号量机制实现这三个进程的互斥与同步活动，并说明所定义的信号量的含义（要求用伪代码描述）。
2. 如下图所示，三个合作进程、、，它们都需要通过同一设备输入各自的数据a、b、c，该输入设备必须互斥地使用，而且其第一个数据必须由进程读取，第二个数据必须由进程读取，第三个数据必须由进程读取（读取后所有进程都可以使用）。然后，三个进程分别只能对输入数据进行下列计算：

最后，进程通过所连接的打印机将计算结果x、y、z的值打印出来。请用信号量实现它们的互斥与同步。

形状

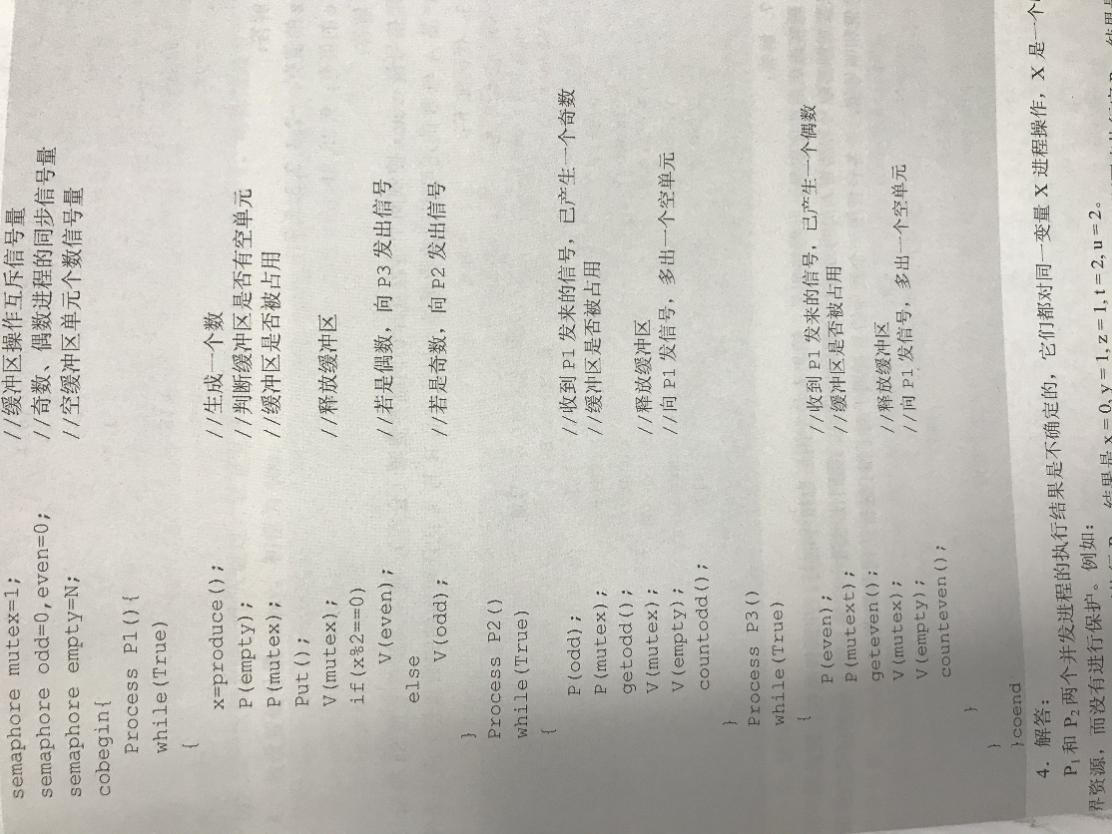
中度可信度描述已自动生成

参考答案：

1. C
2. B

|  |
| --- |
| 1. #include <stdio.h> |
| 1. #include <sys/types.h> |
| 1. #include <signal.h> |
| 1. #include <unistd.h> |
| 1. #include <stdlib.h> |
| 1. #include <sys/wait.h> |
| 1. void sig\_handler() |
| 1. { |
| 1. \_exit(0); |
| 1. } |
|  |
| 1. int main(){ |
| 1. int child\_pid; |
| 1. if((child\_pid=fork())==0){ |
| 1. signal(SIGINT, sig\_handler); |
| 1. while(1); |
| 1. printf("forbidden zone\n"); |
| 1. exit(0); |
| 1. } |
| 1. else{ |
| 1. while(getc(stdin)){ |
| 1. kill(child\_pid, SIGINT); |
| 1. printf("received a signal\n"); |
| 1. wait(0); |
| 1. exit(0); |
| 1. } |
| 1. } |
| 1. } |

4、



5、

