

作业一

作业发布时间：2021/09/19 周日

本次作业要求如下：

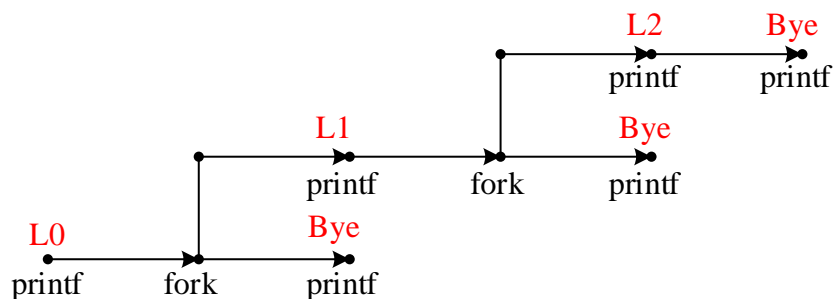
1. 截止日期：2021/09/29 周三晚 24:00
2. 作业提交方式见群文件
3. 命名格式：作业统一命名为“第一次作业+学号+姓名”，作业为 PDF 格式；
4. 注意：
 - (1) 本次作业包括两部分，分别包含 8 个题目和 5 个题目。
 - (2) 选择只需要写答案，大题要求有详细过程，过程算分。
 - (3) 答案请用另一种颜色的笔回答，便于批改，否则视为无效答案。
 - (4) 大题的过程最好在纸上写了拍照，放到 word 里。
5. 本次作业遇到问题请联系群里的助教。

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
2. 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
3. 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
4. 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

5. 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
6. 下列关于线程的叙述中，正确的是（ C ）
- I. 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
 - II. 属于同一个进程的各个线程共享栈空间
 - III. 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
 - IV. 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
7. 判断以下哪个输出是不正确的？（ D ）



- A. L0, Bye, L1, Bye, L2, Bye
 - B. L0, L1, Bye, L2, Bye, Bye
 - C. L0, Bye, L1, L2, Bye, Bye
 - D. L0, Bye, L1, Bye, Bye, L2
8. 分析程序 homework_wait.c，回答下列问题：

```

1.  /* homework_wait.c */
2.  void homework_wait() {
3.      pid_t pid[N];
4.      int i, child_status;
5.      for (i = 0; i < N; i++) {

```

```

6.         if ((pid[i] = fork()) == 0) {
7.             exit(100+i); /* Child */
8.         }
9.     }
10.    printf("hello!\n");
11.    for (i = 0; i < N; i++) { /* Parent */
12.        pid_t wpid = wait(&child_status);
13.        if (WIFEXITED(child_status))
14.            printf("Child %d terminated with exit status %d\n",
15.                wpid, WEXITSTATUS(child_status));
16.        else
17.            printf("Child %d terminate abnormally\n", wpid);
18.    }
19. }

```

- 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。
- 3) 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:

```

1.  /* homework_wait_answer1.c */
2.  void homework_wait() {
3.      pid_t pid[N];
4.      int i, child_status;
5.      for (i = 0; i < N; i++) {
6.          if ((pid[i] = fork()) == 0) {
7.              exit(100+i); /* Child */
8.          }
9.      }
10.     printf("hello!\n");
11.     for (i = 0; i < N; i++) { /* Parent */
12.         pid_t wpid = waitpid(pid[i], &child_status, 0);
13.         if (WIFEXITED(child_status))
14.             printf("Child %d terminated with exit status %d\n",
15.                wpid, WEXITSTATUS(child_status));
16.         else
17.             printf("Child %d terminate abnormally\n", wpid);
18.     }
19. }

```

答案 2:

```
1.  /* homework_wait_answer2.c */
2.  void homework_wait() {
3.      pid_t pid[N], wpid;
4.      int i, child_status;
5.      for (i = 0; i < N; i++) {
6.          if ((pid[i] = fork()) == 0) {
7.              exit(100+i); /* Child */
8.          }
9.      }
10.     printf("hello!\n");
11.
12.     /*回收所有子进程之后再调用 waitpid 就返回-1, 且设置 errno 为 ECHILD */
13.     i = 0;
14.     while((wpid = waitpid(pid[i++], &status, 0)) > 0) {
15.         if (WIFEXITED(child_status))
16.             printf("Child %d terminated with exit status %d\n",
17.                 wpid, WEXITSTATUS(child_status));
18.         else
19.             printf("Child %d terminate abnormally\n", wpid);
20.     }
21.
22.     if (errno != ECHILD)
23.         unix_error("waitpid error");
24.
25.     exit(0);
26. }
```

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（ ）
 - A. 允许其他进程进入与该进程相关的临界区
 - B. 不允许其他进程进入任何临界区
 - C. 允许其他进程抢占处理器，但不得进入该进程的临界区
 - D. 不允许任何进程抢占处理器
2. 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（ ）
 - A. 0, 1
 - B. 1, 0
 - C. 1, 2
 - D. 2, 0
3. 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）

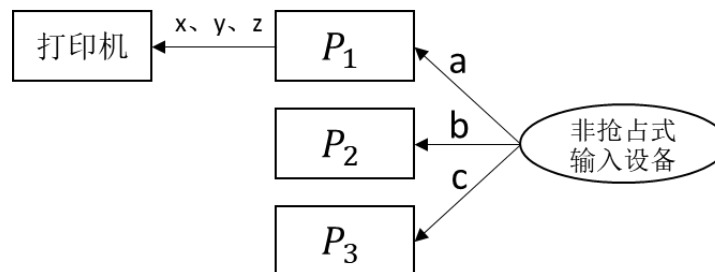
```
1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <signal.h>
4. #include <unistd.h>
5. #include <stdlib.h>
6. #include <sys/wait.h>
7. int main(){
8.     int child_pid;
9.     if((child_pid=fork())==0){
10.         while(1);
11.         printf("forbidden zone\n");
12.         exit(0);
13.     }
14.     else{
15.         while(getc(stdin)){
16.             wait(0);
17.             exit(0);
18.         }
19.     }
```

```
21.    }
22. }
```

4. 三个进程 P_1 、 P_2 、 P_3 互斥使用一个包含 N ($N>0$) 个单元的缓冲区。 P_1 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； P_2 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； P_3 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）。
5. 如下图所示，三个合作进程 P_1 、 P_2 、 P_3 ，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 P_1 进程读取，第二个数据必须由 P_2 进程读取，第三个数据必须由 P_3 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：

$$\begin{aligned} P_1: x &= a + b \\ P_2: y &= a * b \\ P_3: z &= y + c - a \end{aligned}$$

最后， P_1 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。



参考答案：

- 1、 C
2、 B
3、

```
1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <signal.h>
4. #include <unistd.h>
5. #include <stdlib.h>
6. #include <sys/wait.h>
7. void sig_handler()
8. {
9.     _exit(0);
```

```

10. }
11.
12. int main(){
13.     int child_pid;
14.     if((child_pid=fork())==0){
15.         signal(SIGINT, sig_handler);
16.         while(1);
17.         printf("forbidden zone\n");
18.         exit(0);
19.     }
20.     else{
21.         while(getc(stdin)){
22.             kill(child_pid, SIGINT);
23.             printf("received a signal\n");
24.             wait(0);
25.             exit(0);
26.         }
27.     }
28. }

```

4、

```

semaphore mutex=1;           //缓冲区操作互斥信号量
semaphore odd=0,even=0;      //奇数、偶数进程的同步信号量
semaphore empty=N;           //空缓冲区单元个数信号量
cobegin{
    Process P1(){
        while(True)
        {
            x=produce();        //生成一个数
            P(empty);           //判断缓冲区是否有空单元
            P(mutex);           //缓冲区是否被占用
            Put();
            V(mutex);           //释放缓冲区
            if(x%2==0)
                V(even);        //若是偶数，向 P3 发出信号
            else
                V(odd);          //若是奇数，向 P2 发出信号
        }
    }
    Process P2()
    while(True)
    {
        P(odd);                //收到 P1 发来的信号，已产生一个奇数
        P(mutex);               //缓冲区是否被占用
        getodd();
        V(mutex);               //释放缓冲区
        V(empty);               //向 P1 发信号，多出一个空单元
        countodd();
    }
    Process P3()
    while(True)
    {
        P(even);                //收到 P1 发来的信号，已产生一个偶数
        P(mutex);               //缓冲区是否被占用
        geteven();
        V(mutex);               //释放缓冲区
        V(empty);               //向 P1 发信号，多出一个空单元
        counteven();
    }
}
coend

```

5、

为了控制三个进程依次使用输入设备进行输入，需分别设置三个信号量 S_1 、 S_2 、 S_3 ，其中 S_1 的初值为 1， S_2 和 S_3 的初值为 0。使用上述信号量后，三个进程不会同时使用输入设备，故不必再为输入设备设置互斥信号量。另外，还需要设置信号量 S_b 、 S_y 、 S_z 来表示数据 b 是否已经输入，以及 y 、 z 是否已计算完成，它们的初值均为 0。三个进程的动作可描述为：

```
P1() {  
    P(S1);  
    从输入设备输入数据 a;  
    V(S2);  
    P(Sb);  
    x=a+b;  
    P(Sy);  
    P(Sz);  
    使用打印机打印出 x、y、z 的结果;  
}
```

```
P2() {  
    P(S2);  
    从输入设备输入数据 b;  
    V(S3);  
    V(Sb);  
    y=a*b;  
    V(Sy);  
    V(Sy);  
}
```

```
P3() {  
    P(S3);  
    从输入设备输入数据 c;  
    P(Sy);  
    z=y+c-a;  
    V(Sz);  
}
```