操作系统作业 3

范翔宇 PB18000006

1. What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

答：进程间通信的两种模型是信息传递模型和共享内存模型。信息传递对于交换少量数据非常有用（基本用在少量的数据），因为不需要避免冲突。与用于计算机间通信的共享内存相比，它更方便、更容易实现。共享内存可以最大程度地提高通讯速度和便利性，因为共享内存可以在计算机内部以内存传输速度完成。 但是，此方法在共享内存的进程之间的保护和同步上存在折中，而且可能发生冲突。

2. What are the benefits of multi-threading? Which of the following components of program state are shared across threads in a multithreaded process?

   a. Register values

   b. Heap memory

   c. Global variables

   d. Stack memory

答：优点：为进程创建分配内存和资源的成本很高，比创建线程要慢数十倍；进程之间的上下文切换也很昂贵，慢了好几倍；线程可能在不同的内核上并行运行。b c

3. Consider the following code segment:

```
pid t pid;
pid = fork();
if (pid == 0) { /* child process */
    fork();
    thread create( . . .);
}
fork();
```

a. How many unique processes are created?

答：5

b. How many unique threads are created?

答：2

4. The program shown in the following figure uses Pthreads. What would be the output from the program at `LINE C` and `LINE P`?

```
#include <pthread.h>
#include <stdio.h>

int value = 0;
void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
pid_t pid;
pthread_t tid;
pthread_attr_t attr;

   pid = fork();

   if (pid == 0) { /* child process */
      pthread_attr_init(&attr);
      pthread_create(&tid,&attr,runner,NULL);
      pthread_join(tid,NULL);
      printf("CHILD: value = %d",value); /* LINE C */
   }
   else if (pid > 0) { /* parent process */
      wait(NULL);
      printf("PARENT: value = %d",value); /* LINE P */
   }
}

void *runner(void *param) {
   value = 5;
   pthread_exit(0);
}
```

Figure: C program for Question 4.

答：Line C：CHILD: value=5
Line P：PARENT: value=0

5. What are the differences between ordinary pipe and named pipe?

答：普通管道：

①仅用于相关进程（父子关系）

②单向通信

③通信结束后不再存在

命名管道：

①不需要父子关系

②多个进程可以使用命名管道通信

③继续存在直到明确删除

④双向通信

6. What is race condition? Which property can guarantee that race condition will not happen?

答：执行的结果取决于访问共享资源的特定顺序；

进程同步保证了互斥，进而保证竞争条件不会发生

7. The first known correct software solution to the critical-section problem for two processes was developed by Dekker. The two processes, P0 and P1, share the following variables:

```
boolean flag[2]; /* initially false */
int turn;
```

The structure of process Pi (i == 0 or 1) is shown in the following Figure; the other process is Pj (j == 1 or 0). Prove that the algorithm satisfies all

three requirements for the critical-section problem.

```
do {
   flag[i] = true;

   while (flag[j]) {
      if (turn == j) {
         flag[i] = false;
         while (turn == j)
            ; /* do nothing */
         flag[i] = true;
      }
   }

      /* critical section */

   turn = j;
   flag[i] = false;

      /* remainder section */
} while (true);
```

Figure: The structure of process Pi for Question 7.

答：①互斥：当 Pj 在临界区运行时，Pi 想要进入临界区，flag[j]=true,turn=j,Pi 进入 while 循环，直到 turn 变成 i，此时 flag[i]=true，Pi 进入临界区。

②进步：turn 和 flag 只会在临界区中改变，这两个变量控制哪个进程可以执行其临界区。

③有限等待：当一个进程结束时，flag 和 turn 的值会发生变化，等待的进程可能会跳出 while 循环或者不进入 while 循环。

8. Can strict alternation and Peterson's solution sastify all the requirements as a solution of the critical-section problem? Please explain why.

答：严格代替并不能一直满足要求，同一进程不可能连续两次进入临界区，即可能会违反“在其临界区之外运行的任何进程都不应阻止其他进程。”这一要求。Peterson的解决方案可以满足三个要求，但仍存在一

些问题，例如：浪费CPU时间、优先级反转问题等

9. What is semaphore? How to use semaphore to implement section entry and section exit (no busy waiting)? Please give the code.

答：信号量是共享可用资源的数量

Code：

Section entry:

```
void down(semaphore *s) {
    disable_interrupt();
    while ( *s == 0 ) {
        enable_interrupt();
        special_sleep();
        disable_interrupt();
    }
    *s = *s –1;
    enable_interrupt();
}
```

Section exit:

```
void up(semaphore *s) {
    disable_interrupt();
    if ( *s == 0 )   special_wakeup();
    *s = *s + 1;
```

```
        enable_interrupt();
    }
```

10.What is deadlock? List the four requirements of deadlock.

答：死锁是指由于两个或者多个线程互相持有对方所需要的资源，导致这些线程处于等待状态，无法前往执行。

四个要求：

①互斥

②持有和等待

③没有抢占

④循环等待