ICS Seminar Week13 Prep

朱家启 徐梓越 许珈铭

2023.12.17

Rules

```
remainder <- ordinal number in WeChat Group % 4
for all questions do
    if question number % 4 == remainder then
        you should work on it
    end
end
```

19、在 Pthread 线程包使用中，下列代码输出正确的是：（　　）

```
void *th_f(void * arg)
{
   printf("Hello World") ;
   pthread_exit(0) ;
 }
int main(void)
{
   pthread_t tid;
   int st;
   st = pthread_create(&tid,NULL,th_f,NULL);
   if(st<0) {
      printf("Oops, I can not create thread\n");
      exit(-1);
   }
   sleep(1);
   exit(0);
}
```

A. Oops, I can not create thread

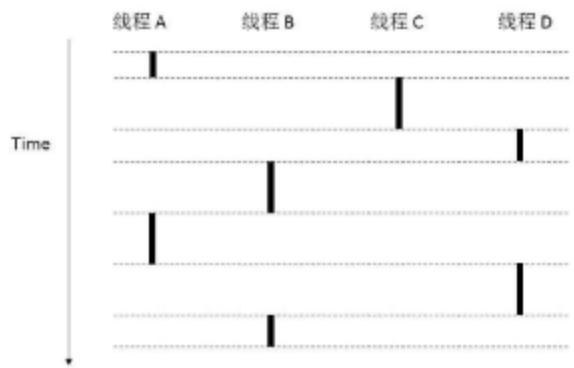
B. Hello World
Oops, I can not create thread

C. Hello World

D. 不输出任何信息

C

24．在一个支持线程的环境下，针对图中所示的场景，下列描述中哪一个是错误的？(　　)



A．线程 A 和线程 D 是并发执行的
B．线程 B 和线程 D 是并发执行的
C．线程 C 和线程 A 是并发执行的
D．线程 B 和线程 C 是并发执行的

C

25. 对于如下 C 语言程序：

```
#include "csapp.h"
void *thread (void * arg)
{
      printf("Hello World") ;
      Pthread_detach(pthread_self()) ;
}
int main(void)
{

      pthread_t tid;
      int sta ;
      sta = Pthread_create(&tid, NULL, thread, NULL);
      if (sta==0)
          printf("Oops, I can not create thread\n");
      exit(NULL);
}
```

在上述程序中，Pthread_detach 函数的作用是(　　)

A. 使主线程阻塞以等待线程 thread 结束

B. 线程 thread 运行结束后会自动释放所有资源

C. 线程 thread 运行后主动释放 CPU 给其他线程

D. 线程 thread 运行后成为僵尸线程

B

26. 两个线程中共享如下一段 C 代码：

```
        for (j = 0;  j < N;  j++)
              count + = 2;
```

假设其对应的汇编代码如下：

```
    movq  (%rdi), %rcx
    testq %rcx,%rcx
    jle   .L2                       Hi
    movl  $0, %eax
.L3:
    movq  count(%rip),%rdx          Li
    addq  $2, %rdx                  Ui
    movq  %rdx, count(%rip)         Si

    addq  $1, %rax
    cmpq  %rcx, %rax                Ti
    jne   .L3
.L2:
```

请问在下列指令顺序对应的轨迹线中，哪一个是安全轨迹线？(　　)

A. H_1, H_2, L_2, L_1, U_2, U_1, S_1, S_2, T_1, T_2

B. H_1, L_1, U_1, H_2, L_2, S_1, T_1, U_2, S_2, T_2

C. H_2, L_2, U_2, H_1, S_2, L_1, T_2, U_1, S_1, T_1

D. H_2, L_2, H_1, L_1, U_1, U_2, S_2, T_2, S_1, T_1

C

19. 有如下代码：

```
int counter = 0;

void * thread(void * vargp)
{
    intthread_var = ((int *) vargp);
    staticintthread_counter = 0;
    thread_internal(thread_var);
    thread_counter ++;
    return NULL;
}
```

```
int main (intargc, const char ** argv)
{
    int tid1, tid2;

    intvar = atoi(argv[1]);

    Pthread_create(&tid1, NULL, thread, (void *)var);
    Pthread_create(&tid2, NULL, thread, (void *)var);

    Pthread_join(tid1, NULL);
    Pthread_join(tid2, NULL);

    return 0;
}
```

则，线程 tid1 与线程 tid2 可以共享的变量是

A. counter, var

B. counter, thread_counter

C. var, thread_counter

D. thread_var, thread_counter

B

20．有四个信号量，初值分别为：a=1，b=1，c=1，d=1。

线程①	线程②	线程③
P(a);	P(d);	P(d);
P(d);	P(a);	P(c);
P(c);	P(c);	P(b);
P(b);	P(b);	P(a);
V(c);	V(d);	V(c);
V(b);	P(d);	V(b);
V(d);	V(a);	V(a);
V(a);	V(b);	V(d);
	V(c);	
	V(d);	

下列哪两个线程并发执行时，一定不会发生死锁？

A．①，②

B．①，③

C．②，③

D．以上选项均不正确

D

19．下列关于“基于进程的并发服务器”的叙述中，哪一个是错误的？

A．父子进程共享文件表

B．父子进程共享文件描述符

C．父子进程不共享用户地址空间

D．进程控制和进程间通信开销大

B

20. 请阅读如下代码：

```
sem_t s;

int main()
{
    int i;
    pthread_t tids[3];
    sem_inti(&s, 0, 1);
    for (i=0; i <3; i++) {
       pthread_create(&tids[i], NULL, justdoit, NULL);
    }
    for (i=0; i <3; i++) {
       pthread_join(&tids[i], NULL);
    }
    return 0;
}
int j=0;

void *justdoit(void *arg)
{
    P(&s);
    j = j + 1;
    V(&s);
    printf("%d\n", j);
}
```

下列哪一个输出结果是不可能的？

A. （1，3，2）

B. （2，3，2）

C. （3，3，2）

D. （2，1，2）

D

19．下列关于死锁的叙述中，不正确的是：

A．所谓死锁是指一组线程被阻塞，等待一个永远不会为真的条件

B．在用信号量及 PV 操作解决生产者消费者问题（多个缓冲区、多个生产者，多个消费者）时，如果先给缓冲区加锁 P(&sp->mutex)，再申请可用的缓冲区槽 P(&sp->slots)；则可能出现死锁

C．printf()是线程安全函数

D．在信号处理函数中使用 printf()不会导致死锁

D

20. 某进程的主线程和对等线程的代码如下所示：

```
sem_t sem;

int main()
{
    int i;
    pthread_t tids[3];
    sem_init(&sem, 0, 1);

    for (i=0; i<3; i++) {
        pthread_create(&tids[i], NULL, thread, NULL);
    }

    for (i=0; i<3; i++) {
        pthread_join(&tids[i], NULL);
    }
    return 0;
}

int y = 15;
void *thread (void *arg)
{
    P(&sem);
    y = y - 3;
    V(&sem);
    printf("%d\n", y);
}
```

执行上述代码后，会产生多少种不同的输出？

A. 11　　B. 12　　C. 13　　D. 14

C

14. 下列关于进程与线程的描述中，哪一个是不正确的？

A. 一个进程可以包含多个线程

B. 进程中的各个线程共享进程的代码、数据、堆和栈

C. 进程中的各个线程拥有自己的线程上下文

D. 线程的上下文切换比进程的上下文切换快

B

15. 给定下列代码片段：

```
char **ptr;  /* global var */
int main(int main, char *argv[]) {
  long i;  pthread_t tid;
  char *msgs[2] = {" Hello from foo", " Hello from bar" };
    ptr = msgs;
    for (i = 0; i < 2; i++)
        Pthread_create(&tid, NULL, thread,(void *)i);
    Pthread_exit(NULL);}

void *thread(void *vargp)
{
  long myid = (long)vargp;
  static int cnt = 0;
  printf("[%ld]:  %s (cnt=%d)\n", myid, ptr[myid], ++cnt);
  return NULL;
}
```

下列哪一组变量集合是对等线程 1 引用的?

A. ptr, cnt, i.m, msgs.m, myid.p0, myid.p1

B. ptr, cnt, msgs.m, myid.p0

C. ptr, cnt, msgs.m, myid.p1

D. ptr, cnt, i.m, msgs.m, myid.p1

C

18. 称一个函数为线程安全的（thread-safe），当且仅当该函数被多个并发线程反复调用时会一直产生正确的结果。下列哪些函数不是线程安全的？

A. 利用 P、V 操作保护了共享变量的函数
B. 保持跨越多个调用的状态的函数
C. 包括了调用者传递存放结果变量的地址的函数
D. 可重入函数

B

19. 在以下基于线程的并发 Echo 服务器代码中，哪一个代码片段会潜在引发不正确的程序执行结果？

A.
```
int main(int argc, char **argv)
{                                   /*假设有配对的”}”
    int listenfd, connfd;
    socklen_t clientlen;
    struct sockaddr_storage clientaddr;
    pthread_t tid;
```
B.
```
listenfd = Open_listenfd(argv[1]);
```
C.
```
while (1) {
    connfd = Accept(listenfd, (SA *) &clientaddr,
&clientlen);
    Pthread_create(&tid, NULL, thread, &connfd);
}
```
D.
```
void *thread(void *vargp)
{
  int connfd = *((int *)vargp);
  Pthread_detach(pthread_self());
  echo(connfd);
  Close(connfd);
  return NULL;
}
```

C

20. 下列关于信号量及 P、V 操作的叙述中，哪一个是不正确的？

A. 信号量是一个非负整数，对信号量只能执行 P 操作和 V 操作

B. 当信号量的值为 0 时，调用 P 操作的线程被挂起

C. P(s)的实现代码中语句：`while (s == 0) wait(); s--;` 应该由内核保证其执行的不可分割性

D. 在保护临界区的解决方案中使用 `sem_wait()`和 `sem_post()`比使用 `pthread_mutex_lock()`和 `pthread_mutex_unlock()`性能好

D

23. 令 L_i 表示加载共享变量到寄存器的指令，U_i 表示更新寄存器的指令，S_i 表示将更新值存回到共享变量的指令，指令 L_1、S_1、U_3、S_3 操作共享变量 cnt，指令 L_2、S_2、L_4、S_4 操作共享变量 num，假定每次只允许交换相邻的两条指令，则将下列指令序列变为正确的至少需要交换多少次：

L_2、L_1、U_1、L_3、L_4、U_4、U_3、S_1、U_2、S_4、S_3、S_2

A.10　　B.11　　C.12　　D.13

C

24. 以下程序在输出有限行之后就终止了，请问最有可能的原因是（假定所有函数都正常执行）

```
#include "csapp.h"
void *thread(void *dummy){
        while (1){
            printf("hello, world!\n");
            Sleep(1);
         }
}
int main(){
        pthread_t tid;
        Pthread_create(&tid, NULL, thread, NULL);
        Sleep(3);
}
```

A. 主线程结束必然引发所有对等线程结束
B. 主线程结束时调用了_exit 导致进程结束
C. 主线程结束后，内核发送 SIGKILL 杀死进程
D. 主线程结束后，内核观察到对等线程运行时间过长，将其杀死

B

25. 有四个信号量，初值分别为：a=1，b=1，c=1，d=1

线程 1：	线程 2：	线程 3：
① P(a);	⑨ P(c);	⑮ P(d);
② P(b);	⑩ P(b);	⑯ P(a);
③ P(d);	⑪ V(c);	⑰ V(d);
④ P(c);	⑫ V(b);	⑱ P(b);
⑤ V(d);	⑬ P(a);	⑲ V(a);
⑥ V(a);	⑭ V(a);	⑳ V(b);
⑦ V(b);		
⑧ V(c);		

上面的程序执行时，下列哪一个执行轨迹执行后已经出现死锁?

A. ⑨①⑩②⑪⑮

B. ①⑨⑩②⑮⑯

C. ⑮①⑨⑩⑯②

D. ①②③④⑨⑮

B

第八题（12 分）

1.请阅读下列代码 badcnt.c：

```
/* Global shared variable */
volatile long cnt = 0; /* Counter */
int main(int argc, char **argv)
{
    long niters;
    pthread_t tid1, tid2;
    niters = atoi(argv[1]);
    Pthread_create(&tid1, NULL, thread, &niters);
    Pthread_create(&tid2, NULL, thread, &niters);
    Pthread_join(tid1, NULL);
    Pthread_join(tid2, NULL);
    /* Check result */
    if (cnt != (2 * niters))
        printf("BOOM! cnt=%ld\n", cnt);
    else
        printf("OK cnt=%ld\n", cnt);
    exit(0);
}
```

```
/* Thread routine */
void *thread(void *vargp)
{
    long i, niters = *((long *)vargp);

    for (i = 0; i < niters; i++)
        cnt++;

    return NULL;
}
```

运行后可能产生如下结果：

```
linux> ./badcnt 10000
OK cnt=20000
```

或者：

```
linux> ./badcnt 10000
BOOM! cnt=13051
linux>
```

为什么会产生不同的结果？请分析产生不同结果的原因。

2.某辆公交车的司机和售票员为保证乘客的安全，需要密切配合、协调工作。司机和售票员的工作流程如下所示。请编写程序，用 P、V 操作来实现司机与售票员之间的同步。

司机进程：

```
while(1) {
    ①
    启动车辆；
    ②
    正常行驶；
    ③
    到站停车；
    ④
}
```

售票员进程：

```
while(1) {
    ⑤
    关门；
    ⑥
    报站名或维持秩序；
    ⑦
    到站开门；
    ⑧
}
```

（1）请设计若干信号量，给出每一个信号量的作用和初值。
（2）请将信号量对应的 PV 操作填写在代码中适当位置。

注：每一标号处可以不填入语句（请标记成 X），或填入一条或多条语句。

标号	对应的操作
①	
②	
③	
④	
⑤	
⑥	
⑦	
⑧	

1．（4 分）

cnt++ 在汇编里会被分成加载、更新、写回三个步骤（2 分）。线程 1 把 cnt 读入寄存器，寄存器加 1，在未写回时，若线程 2 把 cnt 读入自己的寄存器，寄存器加 1，最终两个线程都是把 cnt+1 写入 cnt，这就造成了错误。（2 分）

2．（8 分）

（1）设置两个信号量 s1=0、s2=1（2 分）。s1 用于司机进程是否能启动车辆（1 分），s2 用于售票员进程是否能开门（1 分）。

（2） ① P(s1)

② X

③ X

④ V(s2)

⑤ X

⑥ V(s1)

⑦ P(s2)

⑧ X