1. 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

1.1 多线程编程优点：

▶响应性与多任务执行：如果一个交互程序采用多线程，那么即使部分阻塞或者执行冗长的操作，它仍然可以继续执行，从而增加对用户的响应程度

▶便于数据共享

▶经济：进程创建所需的内存和资源分配非常昂贵。由于线程能共享它们所属进程的资源，所以创建和切换线程更加经济

▶可伸缩性：线程可以在多核处理器上并行运行

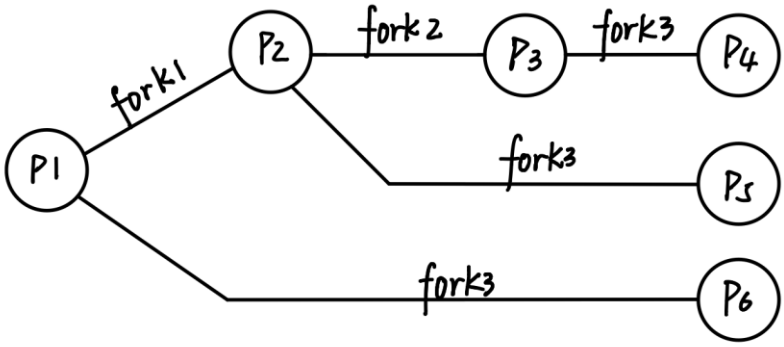
1.2 线程之间共享堆和全局变量。因为堆是在进程空间中开辟出来的，所以肯定是跨线程共享的；同理，全局变量是整个程序所共享的，也应由线程共享。而每个线程都会独立地维护属于自己的寄存器与栈。因此在多线程程序中，线程共享的资源有：b c

2. 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?

b. How many unique threads are created?



2. a 由图可知，一共创建了 6 个进程

2. b P4 P5 分别有两个线程，因此一共有 8 个线程

3. 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);
}
```

子进程会复制父进程的堆、栈、数据段等信息，两者是独立的（在子进程修改全局变量不会影响父进程中的同名全局变量），而进程中的线程会与该进程共享数据段（里面包括全局变量）和堆内存。Value 是全局变量，因此 LINE C 输出是“CHILD: value = 5”，LINE P 输出是“PARENT: value = 0”

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

▶Ordinary pipes are used only for related (parent-child) processes. While parent-child

relationship is not necessary for Named pipes
▶Ordinary Pipes are unidirectional, while named pipes are bidirectional
▶Ordinary Pipes cease to exist after communication has finished, while named pipes continue to exist until it is explicitly deleted.

5. List all the requirements of the entry and exit implementation when solving the critical-section problem. Analyze whether strict alternation satisfies all the requirements.

▶互斥：如果进程Pi 在其临界区内执行，那么其他进程都不能在其临界区执行
▶进步：如果没有进程在临界区执行，并且有进程需要进入临界区，那么只有那些不在剩余区内执行的进程也可以参加选择，以便决定谁能下次进入临界区，而且这种选择不会无限推迟。在临界区外运行的进程不应该阻塞其他进程
▶有限等待：从一个进程做出进入临界区的请求直到这个请求允许为止，其他进程允许进入其临界区的次数具有上限。不会有进程为了进入其临界区而永久等待
▶假定每个进程的执行速度不为零，然而对于 n 个进程的相对速度和 CPU 的数量不做任何假设。解决方案不能依赖进程在临界区内执行的时间
严格轮转没有满足第二条需求

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

死锁是指多个进程在运行过程中因争夺资源而造成的一种僵局，当进程处于这种僵持状态时，若无外力作用，它们都将无法再向前推进。
▶互斥条件：进程要求对所分配的资源进行排它性控制，即在一段时间内某资源仅为一进程所占用。
▶请求和保持条件：当进程因请求资源而阻塞时，对已获得的资源保持不放。
▶不剥夺条件：进程已获得的资源在未使用完之前，不能剥夺，只能在使用完时由自己释放。
▶环路等待条件：在发生死锁时，必然存在一个进程--资源的环形链。

7. What is semaphore? Explain the functionalities of semaphore in process synchronization.

信号量是一种数据类型，它的使用主要是用来保护共享资源，使得资源在一个时刻只有一个进程（线程）所拥有。信号量的值为正的时候，说明它空闲。所测试的线程可以锁定而使用它，并执行 down()使信号量减小。用过后释放资源执行 up()，使信号量增加。若为 0，说明它被占用，请求资源的线程进入睡眠队列中，等待被唤醒。

8. Please use semaphore to provide a deadlock-free solution to address the dining philosopher problem.

可以定义一个小于哲学家人数的信号量来限制同时请求资源的人数，来防止死锁。例如当有 5 个哲学家和 5 支筷子时，如果每个哲学家都拿起自己左边的筷子，那么就睡会发生死锁，但是限制 4 个哲学家 5 支筷子时，就不会发生死锁。

9. Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

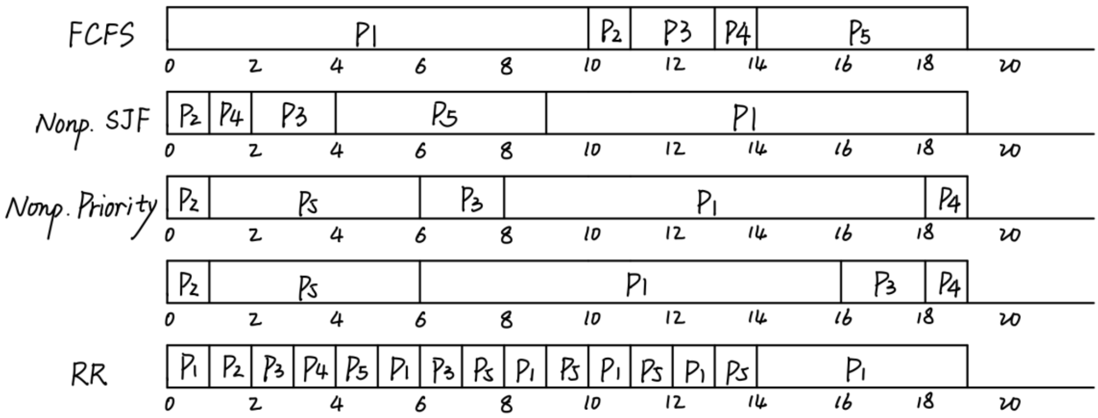
a) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF (nonpreemptive), nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1).

b) What is the turnaround time of each process for each of the scheduling algorithms in part a?

c) What is the waiting time of each process for each of these scheduling algorithms?

d) Which of the algorithms results in the minimum average waiting time (over all processes)?

▶



▶9.b FCFS: P1=10ms P2=1ms P3=2ms P4= 1ms P5=5ms

Nonp.SJF: P1=10ms P2=1ms P3=2ms P4= 1ms P5=5ms

Nonp.Priority: Case1: P1=10ms P2=1ms P3=2ms P4= 1ms P5=5ms

Case2: P1=10ms P2=1ms P3=2ms P4= 1ms P5=5ms

RR: P1=19ms P2=1ms P3=5ms P4=1ms P5=10ms

▶9.c FCFS： P1=0ms P2=10ms P3=11ms P4=13ms P5=14ms

Nonp.SJF: P1=9ms P2=0ms P3=2ms P4=1ms P5=4ms

Nonp.Priority: Case1: P1=8ms P2=0ms P3=6ms P4=18ms P5=1ms

Case2: P1=6ms P2=0ms P3=16ms P4=18ms P5=1ms

RR: P1=9ms P2=1ms P3=5ms P4=3ms P5=9ms

▶9.d FCFS: 9.6ms

Nonp.SJF:3.2ms

Nonp.Priority:Case1: 6.6ms   Case2: 8.2ms

RR: 5.4ms

Nonp.SJF has the shortest waiting time among all the algorithms

10.Which of the following scheduling algorithms could result in starvation?

a) First-come, first-served

b) Shortest job first

c) Round robin

d) Priority

B

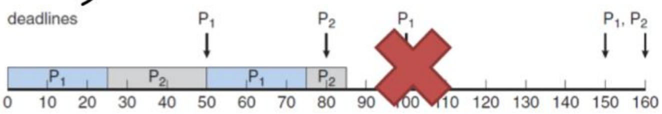
11.Give an example to illustrate under what circumstances ratemonotonic scheduling is inferior to earliest-deadline-first scheduling in meeting the deadlines associated with processes?

Rate Monotonic Scheduling

Example

P1: p1=50, **t1=25**

P2: **p2=80**, t2=35



EDF

Example

P1: p1=50, **t1=25**

P2: **p2=80**, t2=35