操作系统作业一

要求：独立完成，严禁抄袭，按时提交。

提交截至日期：3.13（周日）18:00。

提交方式：提交 PDF 版本，PDF 命名规则为学号_姓名_作业 1，比如 PB200110000_张三_作业 1.pdf。

1. 请分别从系统和用户的角度，阐述操作系统的功能。
2. 概述 multi-programming 和 multi-tasking 的概念及其设计目的。
3. 概述存储的层次以及缓存的思想。
4. 解释什么是系统调用，详细阐述系统调用与 API 的逻辑关系。
5. 阐述 Dual Mode 的工作机制，以及采用 Dual Mode 的原因。
6. 概述操作系统需要提供哪些服务。
7. 分别阐述 Monolithic 单片结构，层次化结构，模块化结构和微内核结构的特点和优劣。
8. 举例说明什么是机制与策略分离的设计原则，并说明该设计的好处。

操作系统作业 2

要求：独立完成，严禁抄袭，按时提交。

提交截至日期：4.10（周日）18:00。

提交方式：提交 PDF 版本，PDF 命名规则为学号_姓名_作业 2， 比如 PB200110000_张三_作业 2.pdf。

1. Including the initial parent process, how many processes are created by the program shown in Figure 1?

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

int main()
{
    int i;

    for (i = 0; i < 4; i++)
       fork();

    return 0;
}
```

Figure 1: Program for Question 1.

2. Explain the circumstances under which the line of code marked printf (“LINE J”) in Figure 2 will be reached.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
       fprintf(stderr, "Fork Failed");
       return 1;
    }
    else if (pid == 0) { /* child process */
       execlp("/bin/ls","ls",NULL);
       printf("LINE J");
    }
    else { /* parent process */
       /* parent will wait for the child to complete */
       wait(NULL);
       printf("Child Complete");
    }

    return 0;
}
```

Figure 2: Program for Question 2.

3. Using the program in Figure 3, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
       fprintf(stderr, "Fork Failed");
       return 1;
    }
    else if (pid == 0) { /* child process */
       pid1 = getpid();
       printf("child: pid = %d",pid); /* A */
       printf("child: pid1 = %d",pid1); /* B */
    }
    else { /* parent process */
       pid1 = getpid();
       printf("parent: pid = %d",pid); /* C */
       printf("parent: pid1 = %d",pid1); /* D */
       wait(NULL);
    }

    return 0;
}
```

Figure 3: Program for Question 3.

4. Using the program shown in Figure 4, explain what the output will be at lines X and Y.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

#define SIZE 5

int nums[SIZE] = {0,1,2,3,4};

int main()
{
int i;
pid_t pid;

   pid = fork();

   if (pid == 0) {
      for (i = 0; i < SIZE; i++) {
         nums[i] *= -i;
         printf("CHILD: %d ",nums[i]); /* LINE X */
      }
   }
   else if (pid > 0) {
      wait(NULL);
      for (i = 0; i < SIZE; i++)
         printf("PARENT: %d ",nums[i]); /* LINE Y */
   }

   return 0;
}
```

Figure 4: Program for Question 4.

5. For the program in Figure 5, will LINE X be executed, and explain why.

```
int main(void) {
     printf("before execl ...\n");
    execl("/bin/ls", "/bin/ls", NULL);
    printf("after execl ...\n");      /*LINE: X*/
    return 0;
}
```

Figure 5: Program for Question 5.

6. Explain why “terminated state” is necessary for processes.
7. Explain what a zombie process is and when a zombie process will be eliminated (i.e., its PCB entry is removed from kernel).
8. Explain what data will be stored in user-space and kernel-space memory for a prosess.
9. Explain the key differences between exec() system call and normal function call.

10.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

11.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?

12.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);
}
```

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

操作系统作业 3

要求：独立完成，严禁抄袭，按时提交。

提交截至日期：4.30（周六）18:00。

提交方式：提交 PDF 版本，PDF 命名规则为学号_姓名_作业 3， 比如 PB200110000_张三_作业 3.pdf。

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

2. For Peterson's solution, prove that it satisfies the three requirements of a solution for the critical section problem.

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

4. Consider the following snapshot of a system:

	Allocation	*Max*
	A B C D	*A B C D*
T_0	1 2 0 2	4 3 1 6
T_1	0 1 1 2	2 4 2 4
T_2	1 2 4 0	3 6 5 1
T_3	1 2 0 1	2 6 2 3
T_4	1 0 0 1	3 1 1 2

Using the banker’s algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. Available = (2, 2, 2, 3)

b. Available = (4, 4, 1, 1)

c. Available = (3, 0, 1, 4)

d. Available = (1, 5, 2, 2)

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

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

7. 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)?

e) Illustrate the pros and cons of the algorithms: FCFS, SJF, priority scheduling and RR.

8. Illustarte the key ideas of rate-monotonic scheduling and earliest-deadline-first scheduling. Give an example to illustrate under what circumstances rate-monotonic scheduling is inferior to earliest-deadline-first scheduling in meeting the deadlines associated with processes?

操作系统作业 4

姓名，学号

1. Explain the following terms：
   Segmentation fault
   TLB
   Page fault
   Demand paging

2. Introduce the concept of thrashing, and explain under what circumstance thrashing will happen.

3. Consider a paging system with the page table stored in memory.
   a. If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?
   b. If we add TLBs, and 75 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 2 nanoseconds, if the entry is present.)

4. Assume a program has just referenced an address in virtual memory. Describe a scenario how each of the following can occur: (If a scenario cannot occur, explain why.)
   - TLB miss with no page fault
   - TLB miss and page fault
   - TLB hit and no page fault
   - TLB hit and page fault

5. Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

6. Consider the following page reference string: 7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0, 1. Assuming demand paging with three frames, how many page faults would occur for the following replacement algorithms?
   - LRU replacement
   - FIFO replacement
   - Optimal replacement

7. Explain what Belady's anomaly is, and what is the feature of stack algorithms which never exhibit Belady's anomaly?

操作系统作业 5

姓名，学号

1. Consider a RAID organization comprising five disks in total, how many blocks are accessed in order to perform the following operations for RAID-5 and RAID-6?
   a. An update of one block of data
   b. An update of seven continuous blocks of data. Assume that the seven contiguous blocks begin at a boundary of a stripe.

2. Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is: 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681 Starting from the current head position, what is the total distance (in cylinders) that the disk armmoves to satisfy all the pending requests for each of the following disk-scheduling algorithms?
   a. FCFS
   b. SSTF
   c. SCAN
   d. LOOK
   e. C-SCAN
   f. C-LOOK

3. Explain what open-file table is and why we need it.

4. What does “755” mean for file permission?

5. Explain the problems of using continuous allocation for file system layout and how to solve them.

6. What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file? What is the major problem of FAT?

7. Consider a file system similar to the one used by UNIX with indexed allocation, and assume that every file uses only one block. How many disk I/O operations might be required to read the contents of a small local file at */a/b/c* in the following two cases? Should provide the detailed workflow.

a. Assume that none of the disk blocks and inodes is currently being cached.
b. Assume that none of the disk blocks is currently being cached but all inodes are in memory.

8. Consider a file system that uses inodes to represent files. Disk blocks are 8-KB in size and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, plus single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

9. What is the difference between hard link and symbolic link?

10. What is the difference between data journaling and metadata journaling? Explain the operation sequence for each of the two journaling methods.

11. What are the three I/O control methods?

12. What services are provided by the kernel I/O subsystem?