12.6

12、进程 P1 通过 fork()函数产生一个子进程 P2。假设执行 fork()函数之前，进程 P1 占用了 53 个（用户态的）物理页，则 fork 函数之后，进程 P1 和进程 P2 共占用_____个（用户态的）物理页；假设执行 fork()函数之前进程 P1 中有一个可读写的物理页，则执行 fork()函数之后，进程 P1 对该物理页的页表项权限为_____。上述两个空格对应内容应该是（　　）

A．53，读写　B．53，只读　C．106，读写　D．106，只读

12、进程 P1 通过 fork()函数产生一个子进程 P2。假设执行 fork()函数之前，进程 P1 占用了 53 个（用户态的）物理页，则 fork 函数之后，进程 P1 和进程 P2 共占用_____个（用户态的）物理页；假设执行 fork()函数之前进程 P1 中有一个可读写的物理页，则执行 fork()函数之后，进程 P1 对该物理页的页表项权限为_____。上述两个空格对应内容应该是（　　）

A．53，读写　B．53，只读　C．106，读写　D．106，只读

答案：选 B。

fork 使用之后，只生成新的 mm_struct 等结构，不会将原来的数据也拷贝一份，而是采用了 copy-on-write 的策略，因而占用的物理内存不会是原来的两倍，同时这两个进程的页表项权限都只有读。

16、已知如下代码段

```
write(fd1, str1, strlen(str1));
write(fd2, str2, strlen(str2));
```

可以在原本为空的文件 ICS.txt 中写下字符串 I love ICS!

对于下面这些对于变量 fd1, fd2, str1, str2 的定义：

(1)

```
int fd1 = open("ICS.txt", O_RDWR);
int fd2 = open("ICS.txt", O_RDWR);
char *str1 = "I love ";
char *str2 = "ICS!";
```

(2)

```
int fd1 = open("ICS.txt", O_RDWR
int fd2 = dup(fd1);
```

```
char *str1 = "I love ";
char *str2 = "ICS!";
```

(3)

```
int fd1 = open("ICS.txt", O_RDWR);
int fd2 = open("ICS.txt", O_RDWR);
char *str1 = "I love ";
char *str2 = "I love ICS!";
```

(4)

```
int fd1 = open("ICS.txt", O_RDWR);
int fd2 = dup(fd1);
char *str1 = "I love ";
char *str2 = "I love ICS!";
```

下面哪一个组合是正确的：（　　）

A. (1)(4)　　B. (2)(3)　　C. (1)(2)(3)(4)　　D. 都不正确

16、已知如下代码段

```
write(fd1, str1, strlen(str1));
write(fd2, str2, strlen(str2));
```

可以在原本为空的文件 ICS.txt 中写下字符串 I love ICS!

对于下面这些对于变量 fd1, fd2, str1, str2 的定义：

(1)

```
int fd1 = open("ICS.txt", O_RDWR);
int fd2 = open("ICS.txt", O_RDWR);
char *str1 = "I love ";
char *str2 = "ICS!";
```

(2)

```
int fd1 = open("ICS.txt", O_RDWR
int fd2 = dup(fd1);
```

答案：B

```
char *str1 = "I love ";
char *str2 = "ICS!";
```

(3)

```
int fd1 = open("ICS.txt", O_RDWR);
int fd2 = open("ICS.txt", O_RDWR);
char *str1 = "I love ";
char *str2 = "I love ICS!";
```

(4)

```
int fd1 = open("ICS.txt", O_RDWR);
int fd2 = dup(fd1);
char *str1 = "I love ";
char *str2 = "I love ICS!";
```

下面哪一个组合是正确的：（　　）

A. (1)(4)　　B. (2)(3)　　C. (1)(2)(3)(4)　　D. 都不正确

Part I

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

```
int main() {
    printf("A\n");
    if (fork() == 0) {
        printf("B\n");
    }
    else {
        printf("C\n");
        A
    }
    printf("D\n");
    exit(0);
}
```

（1）如果程序中的 A 位置的代码为空，列出所有可能的输出结果：（1 分）

Part I

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

```
int main() {
    printf("A\n");
    if (fork() == 0) {
        printf("B\n");
    }
    else {
        printf("C\n");
        A
    }
    printf("D\n");
    exit(0);
```

4 个：分别是 ABDCD ABCDD ACBDD ACDBD（错一个扣半分，多了也扣半分，最多扣 1 分）

Part I

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

```
int main() {
    printf("A\n");
    if (fork() == 0) {
        printf("B\n");
    }
    else {
        printf("C\n");
        A
    }
    printf("D\n");
    exit(0);
}
```

（1）如果程序中的 A 位置的代码为空，列出所有可能的输出结果：（1 分）

（2）如果程序中的 A 位置的代码为：

```
waitpid(-1, NULL, 0);
```

列出所有可能的输出结果：（2 分）

Part I

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

```
int main() {
    printf("A\n");
    if (fork() == 0) {
        printf("B\n");
    }
    else {
        printf("C\n");
        A
    }
    printf("D\n");
    exit(0);
}
```

（2）如果程序中的 A 位置的代码为：

```
waitpid(-1, NULL, 0);
```

列出所有可能的输出结果：（2 分）

3 个：分别是 ABDCD ABCDD ACBDD（每个半分，多了扣一分，最多扣 2 分）

Part I

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

```
int main() {
    printf("A\n");
    if (fork() == 0) {
        printf("B\n");
    }
    else {
        printf("C\n");
        A
    }
    printf("D\n");
    exit(0);
}
```

（3）如果程序中的 A 位置的代码为：

printf("E\n");

列出所有可能的输出结果：（2 分）

7 个：分别是 ABDCED ABCEDD ACEBDD ACEDBD ACBEDD ACBDED ABCDED（错一个扣半分，多了扣一分）

12. 下列这段代码的输出不可能是(　　)

```
void handler()
{
    printf("h");
}

int main()
{
    signal(SIGCHLD, handler) ;

    if ( fork() == 0 ) {
        printf("a") ;
    } else {
        printf("b") ;
    }
    printf("c") ;
    exit(0) ;
}
```

A. abcc　　B. abch　　C. bcach　　D. bchac

12. 下列这段代码的输出不可能是（　　）

```
void handler()
{
    printf("h");
}

int main()
{
    signal(SIGCHLD, handler) ;

    if ( fork() == 0 ) {
        printf("a") ;
    } else {
        printf("b") ;
    }
    printf("c") ;
    exit(0) ;
}
```

【答案】D

【说明】SIGCHLD 信号只有在 fork 的子进程结束时产生，因此 h 只会出现在 ac 之后。

A. abcc　　B. abch　　C. bcach　　D. bchac

13.对于虚拟存储系统，一次访存过程中，下列命中组合不可能发生的是(　　)

A. TLB 未命中，Cache 未命中，Page 未命中

B. TLB 未命中，Cache 命中，Page 命中

C. TLB 命中，Cache 未命中，Page 命中

D. TLB 命中，Cache 命中，Page 未命中

13．对于虚拟存储系统，一次访存过程中，下列命中组合不可能发生的是（　　）

A．　TLB 未命中，Cache 未命中，Page 未命中

B．　TLB 未命中，Cache 命中，Page 命中

C．　TLB 命中，Cache 未命中，Page 命中

D．　TLB 命中，Cache 命中，Page 未命中

【答案】 D

【说明】考察 TLB，Cache，页式虚拟存储器基本性质。

open()'s param

15. ICS.txt 中包含 3000 个字符，考虑如下代码段：

```
int main(int argc, char** argv) {
   int fd = open("ICS.txt", O_CREAT | O_RDWR, S_IRUSR |
S_IWUSR);
   write(fd, "ICS", 3);

   char buf[128];
   int i;
   for (i = 0; i < 10; i++) {
      int fd1 = open("ICS.txt", O_RDWR);
      int fd2 = dup(fd1);

      int cnt = read(fd1, buf, 128);
      write(fd2, buf, cnt);
   }
   return 0;
}
```

上述代码执行完后，ICS.txt 中包含多少个字符（假设所有系统调用都成功）？

（　　）

A. 3　　B. 256　　C. 3000　　D. 3072

open()'s param

15. ICS.txt 中包含 3000 个字符，考虑如下代码段：

```
int main(int argc, char** argv) {
   int fd = open("ICS.txt", O_CREAT | O_RDWR, S_IRUSR |
S_IWUSR);
   write(fd, "ICS", 3);

   char buf[128];
   int i;
   for (i = 0; i < 10; i++) {
      int fd1 = open("ICS.txt", O_RDWR);
      int fd2 = dup(fd1);

      int cnt = read(fd1, buf, 128);
      write(fd2, buf, cnt);
   }
   return 0;
}
```

上述代码执行完后，ICS.txt（　　）

A. 3　　B. 256　　C. 3000　　D. 3072

【答案】C

【说明】主要考查 open 函数的用法。open 不像 fopen，不设置 O_TRUNC 并不会清空文件。所以只会反复把文件中字符 1-128 写到字符 129-256，字符数不变。几个干扰项分别考查 dup 的作用以及 buf 大小对于程序功能的影响。

1.（5 分）以下程序运行时系统调用全部正确执行，buffer.txt 文件的内容为 pekinguniv。请给出代码运行后打印输出的结果，并给出程序运行结束后 buffer.txt 文件的内容。

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

int main() {
    char c;
    int file1 = open("buffer.txt", O_RDWR);
    int file2;

    read(file1, &c, 1);
    file2 = dup(file1);
    write(file2, &c, 1);
    printf("1 = %c\n", c);

    int pid = fork() ;
    if (pid == 0) {
        read(file1, &c, 1);
        write(file2, &c, 1);
        printf("2 = %c\n", c);
        read(file1, &c, 1);
        printf("3 = %c\n", c);
        close(file1);
        exit(0);
    } else {
        waitpid(pid, NULL, 0);
        close(file2);
        dup2(file1, file2);
        read(file2, &c, 1);
        write(file2, &c, 1);
        printf("4 = %c\n", c);
    }
    return 0;
}
```

1.（5 分）以下程序运行时系
pekinguniv。请给出代码
buffer.txt 文件的内容。

答案：

1 = p

2 = k

3 = n

4 = g

buffer.txt 文件内容为 ppkknggniv

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

int main() {
    char c;
    int file1 = open("buffer.txt", O_RDWR);
    int file2;

    read(file1, &c, 1);
    file2 = dup(file1);
    write(file2, &c, 1);
    printf("1 = %c\n", c);

    int pid = fork() ;
    if (pid == 0) {
        read(file1, &c, 1);
        write(file2, &c, 1);
        printf("2 = %c\n", c);
        read(file1, &c, 1);
        printf("3 = %c\n", c);
        close(file1);
        exit(0);
    } else {
        waitpid(pid, NULL, 0);
        close(file2);
        dup2(file1, file2);
        read(file2, &c, 1);
        write(file2, &c, 1);
        printf("4 = %c\n", c);
    }
    return 0;
}
```

2.(5 分)某程序员实现了一个课程实验用的操作系统 ICSNIX,其系统函数 sleep 用以下代码实现。请分析该代码存在哪些问题。

```
1 #include    <signal.h>
2 #include    <unistd.h>
3 static void sig_alrm(int signo)
4 {
5    /* nothing to do, just return to wake up the pause */
6 }
7
8 unsigned int sleep(unsigned int seconds)
9 {
10    if (signal(SIGALRM, sig_alrm) == SIG_ERR)
11       return(seconds);
12
13    alarm(seconds); /* start the timer */
14    pause(); /* next caught signal wakes us up */
15    return(alarm(0)); /* turn off timer, return unslept time */
16}
```

```
if (signal(SIGALRM, sig_alrm) == SIG_ERR)
   return(seconds);

alarm(seconds); /* start the timer */
pause(); /* next caught signal wakes us up */
return(alarm(0)); /* turn off timer, return unslept time */
```

答案:（三个问题若只回答了 1 个或 2 个则每个 2 分，全部回答了得 5 分）

问题 1）由于操作系统调度的原因，alarm 信号触发时，pause 可能还未执行，导致 sleep 调用永不会返回。

问题 2）如果应用程序在调用 sleep 之前已经调用了 alarm，则 sleep 中的 alarm 调用会取消之前设置的 alarm 闹钟。（若用户调用 alarm(5)；sleep(10)；则第 5 秒 sleep 就应该唤醒；若用户调用 alarm(20)；sleep(10)；则 sleep 在 10 秒返回后，再过 10 秒应继续产生一个 SIGALRM 信号。）

问题 3）sleep 的 signal 调用改变了整个程序的 SIGALRM 信号处理方式。因此 sleep 应该保留 signal 的返回值（旧的 SIGALRM 信号处理程序），并在返回前恢复该值。

请阅读下面的代码：

```
 1:  int main(int argc, char** argv) {
 2:      int    fd1    =    open("ICS.txt",    O_CREAT|O_RDWR,
 3:  S_IRUSR|S_IWUSR);
 4:      write(fd1, "abc", 3);
 5:
 6:      int fd2 = fd1;
 7:      int fd3 = dup(fd2);
 8:      int fd4 = open("ICS.txt", O_APPEND|O_RDWR);
 9:      write(fd2, "defghi", 6);
10:      write(fd4, "xyz", 3);
11:
12:      int fd5 = fd4;
13:      dup2(fd3, fd5);
14:      write(fd4, "pqr", 3);
15:
16:      close(fd1);
17:
18:      return 0;
19:  }
```

2.请填写在第 16 行代码刚刚执行完之后，下列变量的值（2 分）

fd1	fd2	fd3	fd4	fd5

请阅读下面的代码：

```
 1:  int main(int argc, char** argv) {
 2:      int    fd1    =    open("ICS.txt",    O_CREAT|O_RDWR,
 3:  S_IRUSR|S_IWUSR);
 4:      write(fd1, "abc", 3);
 5:
 6:      int fd2 = fd1;
 7:      int fd3 = dup(fd2);
 8:      int fd4 = open("ICS.txt", O_APPEND|O_RDWR);
 9:      write(fd2, "defghi", 6);
10:      write(fd4, "xyz", 3);
11:
12:      int fd5 = fd4;
13:      dup2(fd3, fd5);
14:      write(fd4, "pqr", 3);
15:
16:      close(fd1);
17:
18:      return 0;
19:  }
```

2.请填写在第 16 行代码刚刚执行完之后，下列变量的值（2 分）

fd1	fd2	fd3	fd4	fd5
3	3	4	5	5

说明：错 1 空扣 1 分，扣完为止。