秘密★启用前

2021秋ICS小班练习题9
考试时间：50分钟

ECF: Exceptions & Processes
ECF: Signals & Nonlocal Jumps

姓名：　　学号：

答卷说明：
a. 答卷前，考生务必将自己的姓名填写在答题卡指定位置.
b. 答题时，请将答案填写在试卷和答题卡相应位置. 如需改动，请用签字笔将原答案划去，再在规定位置填写修正后的答案. 未在规定区域作答的答案无效.
c. 本卷共4页，卷面分100分. 考试结束后，试卷由助教统一收回.

一、选择题(40分)

(　　)1.关于信号的描述，以下不正确的是_____.
A.在任何时刻，一种类型至多只会有一个待处理信号
B.信号既可以发送给一个进程，也可以发送给一个进程组
C.SIGTERM 和 SIGKILL 信号既不能被捕获，也不能被忽略
D.当进程在前台运行时，键入 Ctrl-C，内核就会发送一个 SIGINT 信号给这个前台进程

(　　)2.下面关于非局部跳转的描述，正确的是_____.
A. siglongjmp函数从env缓冲区中恢复调用环境，触发一个从最近一次初始化env的setjmp调用的返回
B.setjmp 必须放在 main()函数中调用
C.虽然 longjmp 通常不会出错，但仍然需要对其返回值进行出错判断
D.在同一个函数中既可以出现 setjmp，也可以出现 longjmp

(　　)3.下列说法中_____是错误的.
A.中断一定是异步发生的.
B.异常处理程序一定运行在内核模式下.
C.故障处理一定返回到当前指令.
D.陷阱一定是同步发生的

(　　)4.在系统调用成功的情况下，下面哪个输出是可能的?
A.AAB　B.AAA　C.AABB　D.AA

```
int main() {
        intpid = fork();
        if (pid == 0) printf("A");
        else {
                pid = fork();
                if (pid == 0) {
                        printf("A");
                } else {
                printf("B");
                }
        }
        exit(0);
}
```
（第4题图）

(　　)5.以下代码可能的输出为_____.

```
int count = 0;
int pid = fork();
if (pid == 0)printf("count = %d\n",--count);
else printf("count = %d\n",++count);
printf("count = %d\n",++count);
```

A.1 2 -1 0　　B.0 0 -1 1　　C.1 -1 0 0　　D.0 -1 1 2

(　　)6.关于进程，以下说法正确的是:
A.没有设置模式位时，进程运行在用户模式中，允许执行特权指令，例如发起I/O操作。
B.调用 waitpid(-1, NULL, WNOHANG & WUNTRACED)会立即返回：如果调用进程的所有子进程都没有被停止或终止，则返回0；如果有停止或终止的子进程，则返回其中一个的ID。
C.execve 函数的第三个参数 envp 指向一个以 null 结尾的指针数组，其中每一个指针指向一个形如"name=value"的环境变量字符串。
D.进程可以通过使用 signal 函数修改和信号相关联的默认行为，唯一的例外是SIGKILL，它的默认行为是不能修改的。

(　　)7.在系统调用成功的情况下，下列代码会输出_____个hello.

```
void doit(){
        if ( fork() == 0 ) {
                printf("hello\n");
                fork();
        }
        return ;
}
int main(){
        doit();
        printf("hello\n");
        exit(0) ;
}
```

A.3　　B.4
C.5　　D.6

(　　)8.一段程序中阻塞了 SIGCHLD 和 SIGUSR1 信号。接下来，向它按顺序发送SIGCHLD，SIGUSR1，SIGCHLD信号，当程序取消阻塞继续执行时，将处理这三个信号中的哪几个?
A.都不处理
B.处理一次 SIGCHLD
C.处理一次 SIGCHLD，一次 SIGUSR1
D.处理所有三个信号

(　　)9.下列说法正确的是_____.
A.SIGTSTP信号既不能被捕获，也不能被忽略
B.存在信号的默认处理行为是进程停止直到被 SIGCONT 信号重启
C．系统调用不能被中断，因为那是操作系统的工作　　D．子进程能给父进程发送信号，但不能发送给兄弟进程

(　　)10.下面这个程序的输出是_____.

```
volatile long counter = 2;
void handler1(int sig){
        sigset_t mask, prev_mask;
        Sigfillset(&mask);
        Sigprocmask(SIG_BLOCK, &mask,
        &prev_mask); /* Block sigs */
        Sio_putl(--counter);
        Sigprocmask(SIG_SETMASK,
        &prev_mask, NULL); /* Restore sigs */
        _exit(0);
}
```

```
int main(){
        pid_t pid;
        sigset_t mask, prev_mask;
        printf("%ld", counter);
        signal(SIGUSR1, handler1);
        if ((pid = Fork()) == 0) while(1) {};
        Kill(pid, SIGUSR1);
        Waitpid(-1, NULL, 0);
        Sigfillset(&mask);
        Sigprocmask(SIG_BLOCK, &mask,
        &prev_mask); /* Block sigs */
        printf("%ld", ++counter);
        Sigprocmask(SIG_SETMASK,
        &prev_mask, NULL); /* Restore sigs */
        exit(0);
}
```

A.13　　B.213　　C.212　　D.12

二、非选择题(60分)

11(20分,每空4分). 以下程序运行时系统调用全部正确执行，且每个信号都被处理到。请给出代码运行后所有可能的2种输出结果。

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
int c = 1;

void handler1(int sig) {   c++;      printf("%d", c);}
```

//见下页

```
int main() {
        signal(SIGUSR1, handler1);
        sigset_t s;
        sigemptyset(&s);
        sigaddset(&s, SIGUSR1);
        sigprocmask(SIG_BLOCK, &s, 0);
        intpid = fork()?fork():fork();
        if (pid == 0) {
                kill(getppid(), SIGUSR1);
                printf("S");
                sigprocmask(SIG_UNBLOCK, &s, 0);
                exit(0);
        }
                else {
                while (waitpid(-1, NULL, 0) != -1);
                sigprocmask(SIG_UNBLOCK, &s, 0);
                printf("P");
        }
        return 0;
}
```

答: _____, _____,

12(40分,每空2分).埃氏筛 (Sieve of Eratosthenes) 是一种简单且历史悠久的筛选一定范围内所有素数的方法。其原理是从2开始，将每个素数的倍数，标记成合数，然后筛去。那么剩下的数中最小的必定是素数，可以用作新的筛子。例如，寻找[2,10]中的素数：

```
列出所有数          2 3 4 5 6 7 8 9 10
2 为筛子，余下        3   5   7   9
3 为筛子，余下            5   7
5 为筛子，余下                7
7 为筛子，余下空，结束
```

数所求范围内包含素数 2 3 5 7

下面是 Alice 基于进程控制实现的埃氏筛算法。但是她在实现时遇到了不少问题，她请你帮忙解决。**在本题中，你可以认为所有的函数调用都能正确返回，并且程序的标准输出是命令行。**

额外信息：

1. Linux 中有一类特殊文件叫做管道，一个管道有两个文件描述符，其中一个专门用于写，一个专门用于读。写进去的内容就像通过水管一样可以顺次地从读端口被读取。在 c 中，可以使用 int pipe(int *p) 打开一个管道文件。正确打开则 p[0] 是专用于读的端口，p[1] 是专用于写的端口。管道的读写及关闭操作都和普通文件一样。如果使用 write 写管道时文件的读端口已经关闭，那么进程会收到一个 SIGPIPE 信号，并触发 broken pipe 错误。如果使用 read 读管道时文件的写端口已经关闭，那么可以正常读取内容直到管道清空，这时再对管道 read 将返回 EOF。
2. 代码中首字母大写的函数均仿照课本做了 Stevensen 风格的错误检查的封装。其中 int WriteInt(int fd, int val); 每次根据文件描述符向文件中按照 int 格式写入 val 值， int ReadInt(int fd, int *ptr) 每次根据文件描述符从文件中读取一个 int 的内容到 ptr 指向的单元中。

Part A. Alice 对代码的几个细节还没有考虑清楚，请你帮助她完善细节。每个空只需要填一个 c 的关键字/数值常量/变量名。已知(1)(2)(4)(5)(6)(7)(8)为数值常量，(3)(9)为变量，(10)为关键字。

```
1.  int main() {
2.      int pp[2], save;
3.      /* feed every int in [2,9999] into the pipe */
4.      Pipe(pp);
5.      for (int i = 2; i < 10000; ++i)
6.        WriteInt(pp[1], i);
7.      Close(pp[(1)____]);                  // close the write end
8.
9.      int prime = 0, mark = (2)______;
10.     while (ReadInt(pp[0], &save)) {
11.       if (!prime) {
12.         prime = save;                    // current sieve
13.         printf("%d ", prime);
14.       } else if (save % (3)_____) {      // not a multiple
15.         if (!mark) {
16.           int tmp[2];
17.           Pipe(tmp);
18.           if (Fork()) {
19.             Close(tmp[0]);               // close the read end
20.             pp[1]= tmp[(4)___];
21.             mark ^= 1;
22.           } else {
23.             Close(pp[(5)____]);
24.             Close(tmp[(6)____]);
25.             pp[(7)____] = tmp[(8)___];
26.             (9)_______ = 0;
27.             (10)___________;
28.           }
29.         }
30.         WriteInt(pp[1], save);           // feed to the child
31.       }
32.     }
33.     Close(pp[0]);
34.     Wait(NULL);                          // reap the child
35.   }
```

Part B. Alice 很感谢你的帮助！她迫不及待地编译，并顺利通过了。但是她很纳闷自己的代码没有任何输出，程序也结束不了，一连试了几次都是这样。你看了代码后，含蓄地提醒她好像有文件没有及时关闭，导致第 10 行的 ReadInt 并不返回 EOF。Alice 听到后，立即在第_____行后添加了一条语句_____。

Part C. Alice 非常感激你，现在程序能正常输出所有素数了。可是，Alice 发现很多素数输出了不止一遍，输出的顺序每次运行时也都不完全一样。请你简明扼要地解释原因？

答：_____.

为了解决这个问题，你认为**最早**可以在_____行后增加一条语句_____。同时，你认为这样修改过后输出的顺序是否唯一？_____（填 是/否）。

Part D. Alice 发现如果把搜索范围从[2,9999]改到[2,19999]那么程序总是运行失败。请你简要分析**最主要的**原因?

答:_____.

Part E. 针对 Part D. 中你发现的问题，你决定只在某一行上**增加一个字符**。立刻程序不仅能正确执行输出解果，而且速度**显著提升**。那么你将第_____行改为_____。进一步分析，如果这时搜索范围扩大为 [2,19999] 那么程序运行是否会失败？_____（填 是/否/不确定）。

2021秋ICS小班18班第(9)次考试
考试时间：50分钟

参考答案:
一、选择题（40分） 1-5: **CDCAA** 6-10: **CBCBA**
(第10题注意printf的格式化字符串没有换行符，_exit(status)不刷新缓冲区
二、非选择题
11（20分）： **S2PS2P** **SS2P2P**
12（40分）： **代码地址：** https://pastebin.ubuntu.com/p/kBjb3vccQF/

一. 埃氏筛（Sieve of Eratosthenes）是一种简单且历史悠久的筛选一定范围内所有素数的方法。其原理是从2开始，将每个素数的倍数，标记成合数，然后筛去。那么剩下的数中最小的必定是素数，可以用作新的筛子。下面是 Alice 基于进程控制实现的埃氏筛算法。但是她在实现时遇到了不少问题，她请你帮忙解决。**在本题中，你可以认为所有的函数调用都能正确返回，并且程序的标准输出是命令行。**

额外信息：

1. Linux 中有一类特殊文件叫做管道，一个管道有两个文件描述符，其中一个专门用于写，一个专门用于读。写进去的内容就像通过水管一样可以顺次地从读端口被读取。在 c 中，可以使用 int pipe(int *p) 打开一个管道文件。正确打开则 p[0] 是专用于读的端口，p[1] 是专用于写的端口。管道的读写及关闭操作都和普通文件一样。如果使用 write 写管道时文件的读端口已经关闭，那么进程会收到一个 SIGPIPE 信号，并触发 broken pipe 错误。如果使用 read 读管道时文件的写端口已经关闭，那么可以正常读取内容直到管道清空，这时再对管道 read 将返回 EOF。
2. 代码中首字母大写的函数均仿照课本做了 Stevensen 风格的错误检查的封装。其中 int WriteInt(int fd, int val); 每次根据文件描述符向文件中按照 int 格式写入 val 值， int ReadInt(int fd, int *ptr) 每次根据文件描述符从文件中读取一个 int 的内容到 ptr 指向的单元中。

Part A. Alice 对代码的几个细节还没有考虑清楚，请你帮助她完善细节。每个空只需要填一个 c 的关键字/数值常量/变量名。已知(1)(2)(4)(5)(6)(7)(8)为数值常量，(3)(9)为变量，(10)为关键字。

Part B. Alice 很感谢你的帮助！她迫不及待地编译，并顺利通过了。但是她很纳闷自己的代码没有任何输出，程序也结束不了，一连试了几次都是这样。你看了代码后，含蓄地提醒她好像有文件没有及时关闭，导致第 10 行的 ReadInt 并不返回 EOF。Alice 听到后，立即在第_32/33_行后添加了一条语句____if (mark) Close(pp[1]);_。

Part C. Alice 非常感激你，现在程序能正常输出所有素数了。可是，Alice 发现很多素数输出了不止一遍，输出的顺序每次运行时也都不完全一样。请你简明扼要地解释原因？

标准输出使用行缓冲区，在 fork 出新进程时，缓冲区的内容一起被复制了。最后刷新到命令行就会出现重复的内容。

为了解决这个问题，你认为**最早**可以在__10_行后增加一条语句___fflush(stdout)____。同时，你认为这样修改过后输出的顺序是否唯一？_是_（填 是/否）。

Part D. Alice 发现如果把搜索范围从[2,9999]改到[2,19999]那么程序总是运行失败。请你简要分析**最主要的**原因？

进程资源是有限的。如果有 N 个素数，由于每一次一个数必须要从父进程依次的经过 8 个子进程直到在某层被筛掉，那么最后一个素数被筛除时，该代码此时一共实例化了 N 个活跃进程，他们每一个都持有文件资源，也占据 PCB 的资源。

代码的进程结构：

sieve for 2 -> sieve for 3 -> sieve for 5 -> ... -> sieve for p_N

管道结构

sieve for 2 -- sieve for 3 -- sieve for 5 -- ... -- sieve for p_N

最后一个素数的传递途径：

从根进程开始依次通过各个管道直到 p_N。

a -> b 代表 b 是 a 的子进程。

```
1.  int main() {
2.     int pp[2], save;
3.     /* feed every int in [2,9999] into the pipe */
4.     Pipe(pp);
5.     for (int i = 2; i < 10000; ++i)
6.       WriteInt(pp[1], i);
7.     Close(pp[(1)_1__]);                  // close the write end
8.
9.     int prime = 0, mark = (2)__0___;
10.    while (ReadInt(pp[0], &save)) {
11.      if (!prime) {
12.        prime = save;                    // current sieve
13.        printf("%d ", prime);
14.      } else if (save % (3)prime) { // not a multiple
15.        if (!mark) {
16.          int tmp[2];
17.          Pipe(tmp);
18.          if (Fork()) {
19.            Close(tmp[0]);              // close the read end
20.            pp[1] = tmp[(4)1];
21.            mark ^= 1;
22.          } else {
23.            Close(pp[(5)0]);
24.            Close(tmp[(6)1]);
25.            pp[(7)0] = tmp[(8)0];
26.            (9)prime = 0;
27.            (10)continue;
28.          }
29.        }
30.        WriteInt(pp[1], save);          // feed to the child
31.      }
32.    }
33.    Close(pp[0]);
34.    if (mark) Close(pp[1]); (or after line 32) // Part B
35.    Wait(NULL);                         // reap the child
36.  }
```

Part E 针对 Part D. 中你发现的问题，你决定只在某一行上**增加一个字符**。立刻程序不仅能正确执行输出解果，而且速度显著提升。那么你将第___18__行改为 _____if(!Fork())___。进一步分析，如果这时搜索范围扩大为 [2,19999] 那么程序运行是否会失败？___不确定__（填 是/否/不确定）。

新的进程结构如下：

Print
- ├── sieve for 2
- ├── sieve for 3
- ├── ...
- └── sieve for p_{N-1} [这个细节无关紧要，Dirichlet定理保证a>=2时[a,2a]一定有素数，所以埃氏筛到最后一个素数时筛剩下的一定只有它一个数了]

管道结构（一种可能）：

sieve for 2 -- sieve for 3 -- sieve for 5 -- ... -- sieve for p_{N-1} -- Print

由于Print始终需要从新分配的子进程中读数，子进程将更有可能占据时间片并执行完成并成为僵死进程。尽管父进程还未回收导致其文件资源仍然占用，同时它还占据进程表的表项，但因为实际系统的资源使用短期来看变数较大，不能做断言说一定能/不能正常执行。