Lab05 - Assignment - Answer

1. 进程

1.1 基础

请写这样一个程序（不是函数）：传入三个参数，传入该程序的第一个参数用以判断该程序进行有理数算术加运算还是减运算（0 表示将要进行加运算，1 表示将要进行减运算），第二个第三个参数分别是加（减）运算的第一第二个元素。（提示：`main(int argc, char* argv[])`，可获取命令行参数）。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc,char * argv[])
{
    int a,b,c;
    a=atoi(argv[1]);
    b=atoi(argv[2]);
    c=atoi(argv[3]);
    if(a==0) {
        printf("%d",b+c);
    } else if(a==1) {
        printf("%d",b-c);
    }
}
```

1.2 僵尸进程

僵尸进程有什么危害？编写一个会产生僵尸进程的程序并运行，在终端查看当前进程。然后利用终端杀死该进程。

> 僵尸进程的危害：僵尸进程的危害：僵尸进程虽然不会占用很多资源，但它存在于系统的任务列表，而系统能使用的进程号是有限的，如果产生大量僵尸进程，将会因为无进程号可用而导致系统无法产生新的进程。

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <fcntl.h>

int main()
{
    do{
        sleep(60);
        if(fork()){
```

```
        }else{

        }
    }while(1);
}
```

```
root@DESKTOP-QAENTOG:~/SystemProgram/lab5# ./zombie&
[1] 21649
root@DESKTOP-QAENTOG:~/SystemProgram/lab5# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0    896   532 ?        Sl   16:10   0:00 /init
root        13  0.0  0.0    896    88 ?        Ss   16:10   0:00 /init
root        14  0.0  0.0    896    88 ?        S    16:10   0:00 /init
root        15  0.0  0.0   2612   592 pts/0    Ss+  16:10   0:00 sh -c "$VSC
root        16  0.0  0.0   2612   528 pts/0    S+   16:10   0:00 sh /mnt/c/Us
root        21  0.0  0.0   2612   528 pts/0    S+   16:10   0:00 sh /root/.vs
root        25  0.4  3.8 921004 78604 pts/0    Rl+  16:10   0:17 /root/.vscoo
root        36  0.1  3.1 651796 62680 pts/0    Rl+  16:10   0:04 /root/.vscoo
root        65  0.0  0.0    904    88 ?        Ss   16:10   0:00 /init
root        66  0.0  0.0    904    96 ?        S    16:10   0:01 /init
root        67  0.1  2.3 589088 47268 pts/1    Ssl+ 16:10   0:03 /root/.vscoo
root        74  0.0  0.0    904    88 ?        Ss   16:10   0:00 /init
root        75  0.1  0.0    904    96 ?        S    16:10   0:04 /init
root        76  0.2  2.5 596036 52240 pts/2    Ssl+ 16:10   0:09 /root/.vscoo
root        83  2.0  8.9 1024332 181424 pts/0  Sl+  16:10   1:12 /root/.vscoo
root        94  0.1  2.7 835504 55664 pts/0    Sl+  16:10   0:04 /root/.vscoo
root       202  0.0  0.2   9196  4336 pts/3    Ss+  16:10   0:00 /usr/bin/bas
root       333  0.8  3.1 1884524 64228 pts/0   Sl+  16:11   0:31 /root/.vscoo
root      8172  0.0  2.2 589724 44592 pts/0    Sl+  16:43   0:00 /root/.vscoo
root      9049  0.0  0.2   9196  4340 pts/4    Ss   16:46   0:00 /usr/bin/bas
root     20973  0.3  1.1 5244512 23908 pts/0   Sl+  17:09   0:00 /root/.vscoo
root     21649  0.0  0.0   2364   572 pts/4    S    17:11   0:00 ./zombie
root     21683  0.0  0.1  10620  3236 pts/4    R+   17:11   0:00 ps aux
root@DESKTOP-QAENTOG:~/SystemProgram/lab5# kill 21649
[1]+  Terminated              ./zombie
root@DESKTOP-QAENTOG:~/SystemProgram/lab5#
```

2. 进程控制

2.1 fork 与 wait

编写一段C程序，由父进程创建两个子进程，父进程打印字符`B`，两个子进程分别打印`A`和`C`，并且要求最终的输出为`ABC`。

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <fcntl.h>
```

```c
int main()
{
    int pid;
    if ((pid=fork())> 0)
    {
        wait(NULL);
        putchar('B');
        fflush(stdout);
        if(fork()==0){
            putchar('C');
        }
    }
    else
    {
        putchar('A');
        exit(0);
    }
}
```

2.2 exec族函数

请编写一段C程序，该程序调用调用exec族函数，对当前目录使用`ls -l`。

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <fcntl.h>

int main()
{
    execl("/bin/ls","ls", "-l",NULL);
}
```

2.3 综合应用

请你编写一段C程序，该程序按顺序依次调用题目1.1中的可执行文件分别执行1+1与1-1，随后调用/bin/rm删除题目1.1的可执行文件。

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <fcntl.h>

int main()
{
```

```
    if(fork()>0){
        wait(NULL);
        if(fork()>0){
            wait(NULL);
            if(fork()>0){
                wait(NULL);
            }else{
                execl("/bin/rm","rm","a",NULL);
            }
        }else{
            execl("./a","a", "1","1","1",NULL);
        }
    }else{
        execl("./a","a", "0","1","1",NULL);
    }
}
```

3. 进程通信

3.1 重定向

请编写一个C程序，使用文件描述符与重定向把你的学号输出到`student.txt`文件中

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include <unistd.h>
int main()
{
    int fd=open("student.txt",O_RDWR);
    close(1);
    dup(fd);
    printf("19377251");
}
```

3.2 综合应用

请使用管道编写素数筛选的并发版本C程序`primes.c`，程序原理见下面的介绍或者查看这个网站。该程序读入命令行的第一个参数n，随后输出1-n之间的素数。

运行命令

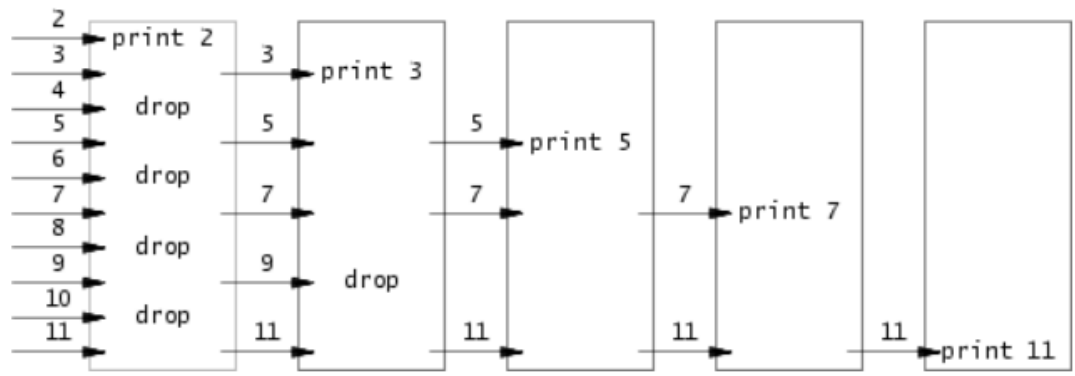
```
gcc primes.c -o primes
./primes 7
```

输出为

```
2
3
5
7
```

考虑所有小于1000的素数的生成。Eratosthenes的筛选法可以通过执行以下伪代码来模拟：

```
p = get a number from left neighbor
print p
loop:
 n = get a number from left neighbor
 if (p does not divide n)
     send n to right neighbor
p = 从左邻居中获取一个数
print p
loop:
 n = 从左邻居中获取一个数
 if (n不能被p整除)
     将n发送给右邻居
```



生成进程可以将数字2、3、4、…、1000输入管道的左端：行中的第一个进程消除2的倍数，第二个进程消除3的倍数，第三个进程消除5的倍数，依此类推。

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <fcntl.h>

void prime(int p[2])
{
    int np[2];
    int number = 0;
    int pr = 0;
    int pass = 0;
    pipe(np);
    while (1) {
        int status = read(p[0], &number, 4);
```

```
        if (status == 0)
            break;
        if (pr == 0) {
            pr = number;
            printf("%d\n", number);
        } else if (number % pr) {
            pass = 1;
            write(np[1], &number, 4);
        }
    }
    close(p[0]);
    close(np[1]);
    if (pass) {
        int pid = fork();
        if (pid == 0) {
            prime(np);
        } else {
            close(np[0]);
            wait(NULL);
        }
    }
}
int main(int argc, char *argv[])
{
    int n;
    n = atoi(argv[1]);
    int p[2];
    pipe(p);
    for (int i = 2; i <= n; i++) {
        write(p[1], &i, 4);
    }
    close(p[1]);
    int pid = fork();
    if (pid == 0) {
        prime(p);
    } else {
        close(p[0]);
        wait(NULL);
    }
}
```