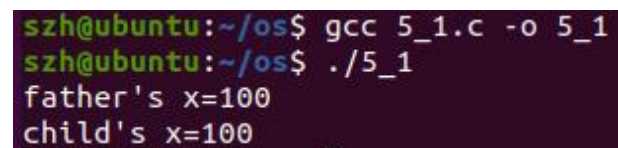


## 5.1

编写一个调用 `fork()` 的程序。在调用 `fork()` 之前，让主进程访问一个变量（例如 `x`）并将其值设置为某个值（例如 100）。子进程中的变量有什么值？当子进程和父进程都改变 `x` 的值时，变量会发生什么？

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

int main(int argc, char *argv[]){
    int x=100;
    int rc=fork();
    if(rc<0){
        printf("error");
        exit(1);
    }
    else if(rc==0){///child
        printf("child's x=%d\n",x);
    }
    else{///father
        printf("father's x=%d\n",x);
    }
    return 0;
}
```



```
szh@ubuntu:~/os$ gcc 5_1.c -o 5_1
szh@ubuntu:~/os$ ./5_1
father's x=100
child's x=100
```

两者返回结果都是 100

```

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

int main(int argc,char *argv[]){
    int x=100;
    int rc=fork();
    if(rc<0){
        printf("error");
        exit(1);
    }
    else if(rc==0){///child
        x=-1;
        printf("child's x=%d\n",x);
    }
    else{///father
        x=10;
        printf("father's x=%d\n",x);
    }
    return 0;
}

```

```

szh@ubuntu:~/os$ gcc 5_1.c -o 5_1
szh@ubuntu:~/os$ ./5_1
father's x=10
child's x=-1

```

修改之后则各自变成各自修改后的的 x

## 5.3

使用 fork()编写一个程序。子进程应打印“hello”，父进程应打印“goodbye”。你应该尝试确保子进程始终先打印。你能否不在父进程调用 wait()而做到这一点呢？

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>
int main(int argc, char *argv[]){
    int rc=fork();
    if(rc<0){
        printf("error");
        exit(1);
    }
    else if(rc==0){//child
        printf("hello\n");
    }
    else{//father
        sleep(1);//执行挂起1ms
        printf("goodbye\n");
    }
    return 0;
}
```

```
szh@ubuntu:~/os$ gcc 5_3.c -o 5_3
szh@ubuntu:~/os$ ./5_3
hello
goodbye
```

程序执行时会先打印 hello，一段时间后 goodbye 被打印出

## 5.5

现在编写一个程序，在父进程中使用 wait()，等待子进程完成。wait()返回什么？如果你在子进程中使用 wait()会发生什么？

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>
int main(int argc, char *argv[]){
    int rc=fork();
    if(rc<0){
        printf("error");
        exit(1);
    }
    else if(rc==0){//child
        int res=wait(NULL);
        printf("child %d %d \n",res,(int)getpid());
        printf("child %d\n",(int)getpid());
    }
    else{//father
        int res= wait(NULL);
        printf("father %d %d\n",res,(int)getpid());
        printf("father %d\n",(int)getpid());
    }
    return 0;
}
```

```
szh@ubuntu:~/os$ gcc 5_5.c -o 5_5
szh@ubuntu:~/os$ ./5_5
child 2542
father 2542 2541
```

父进程等待子进程完成，返回子进程的 pid

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>
int main(int argc, char *argv[]){
    int rc=fork();
    if(rc<0){
        printf("error");
        exit(1);
    }
    else if(rc==0){//child
        int res=wait(NULL);
        printf("child %d %d \n",res,(int)getpid());
        printf("child %d\n",(int)getpid());
    }
    else{//father
        int res= wait(NULL);
        printf("father %d %d\n",res,(int)getpid());
        printf("father %d\n",(int)getpid());
    }
    return 0;
}
```

```
szh@ubuntu:~/os$ gcc 5_5.c -o 5_5
szh@ubuntu:~/os$ ./5_5
father 2484
child -1 2485
```

子进程等待父进程完成，返回的是-1

## 5.7

编写一个创建子进程的程序，然后在子进程中关闭标准输出（STDOUT\_FILENO）。如果子进程在关闭描述符后调用 printf() 打印输出，会发生什么？

```
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>
int main(int argc, char *argv[]){
    int rc=fork();
    if(rc<0){
        printf("error");
        exit(1);
    }
    else if(rc==0){//child
        close(STDOUT_FILENO);
        printf("child %d \n",(int)getpid());
    }
    else{//father
        printf("father %d\n",(int)getpid());
    }
}
return 0;
}
```

```
szh@ubuntu:~/os$ gcc 5_7.c -o 5_7
szh@ubuntu:~/os$ ./5_7
father 2660
```

只会输出父程序的 printf