**实验二 进程控制**

**实验内容一：**

一、需求分析和描述

采用系统调用fork()，编写一个C程序，以便在子进程中生成这个序列，要求：

(1)从命令行中提供启动数字

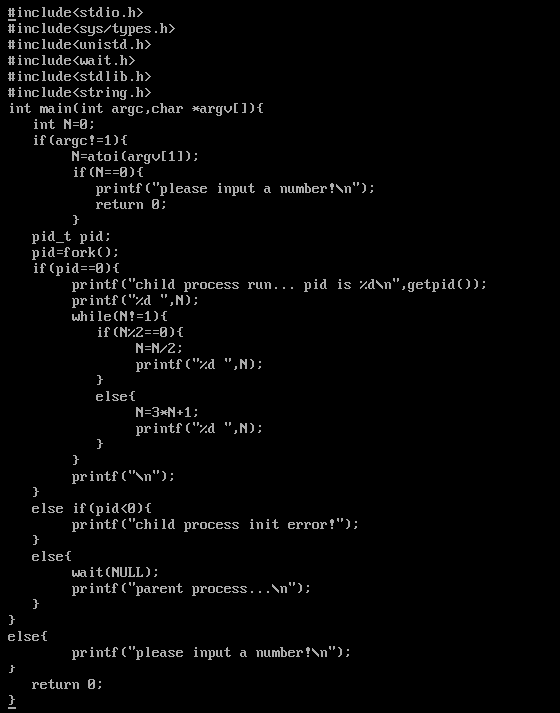
(2)由子进程输出数字序列

(3)父进程等子进程结束后再退出

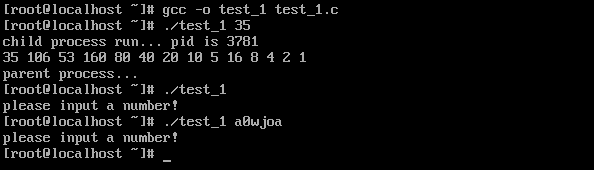
二、实验设计

首先判断输入的是否是数字，然后由父进程fork()出子进程，由子进程计算并输出

三、编码实现



四、运行截图



五、运行结果分析

子进程计算并输出序列，父进程等待子进程结束后退出

**实验内容二：**

一、需求分析和描述

以共享内存技术编程实现Collatz猜想，

要求在父子进程之间建立一个共享内存对象，允许子进程将序列内容写入共享内存对象，当子进程完成时，父进程输出序列

父进程包括如下步骤：  
 建立共享内存对象(shm\_open(),ftruncate(),mmap())

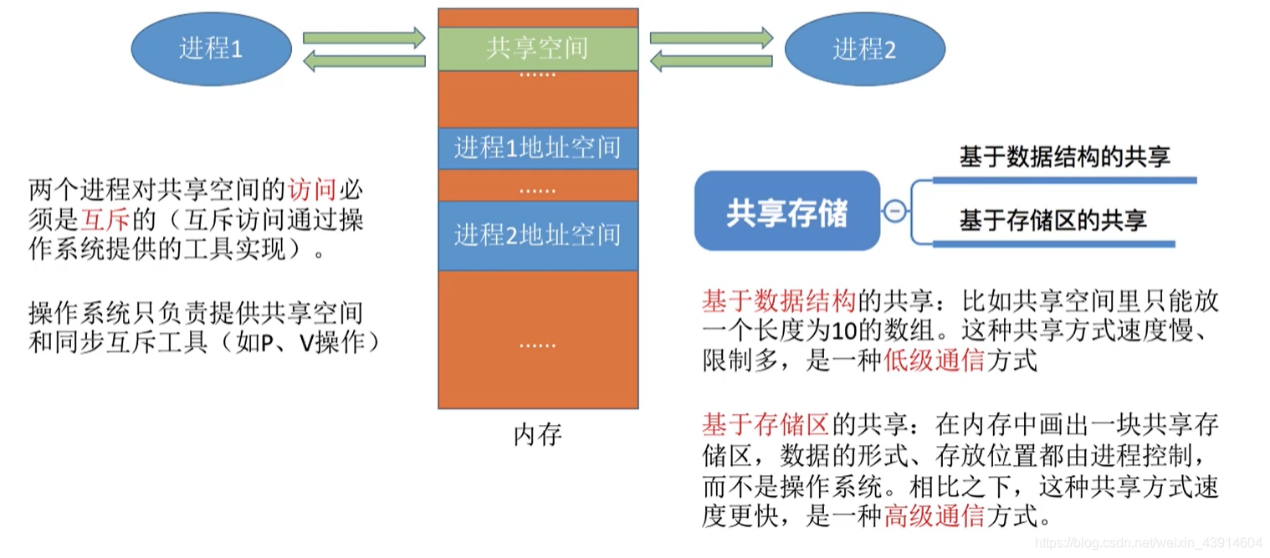
建立子进程并等待他终止

输出共享内存的内容

删除共享内存对象

二、实验设计

共享内容是进程之间通信的一种方式



* **shm\_open和shm\_unlink函数**

shm\_open()用于创建一个新的Posix共享内存对象或打开一个已存在的Posix共享内存对象。

shm\_unlink()用于从系统中删除一个Posix共享内存对象。

**int shm\_open(const char \*name, int oflag, mode\_t mode);**

//成功返回非负标识符，失败返回-1

**int shun\_unlink(const char \*name);**

//成功返回0，失败返回-1

shm\_open参数说明：

* oflag参数不能设置O\_WRONLY标志
* 和mq\_open、sem\_open不同，shm\_open的mode参数总是必须指定，当指定了O\_CREAT标志时，mode为用户权限位，否则将mode设为0

shm\_open的返回值是一个描述符，它随后用作mmap的第五个参数fd。

* **ftruncate函数**

处理mmap的时候，普通文件或Posix共享内存对象的大小都可以通过调用ftruncate设置。在使用时需要包含头文件，#include <unistd.h>

**int ftruncate(int fd, off\_t length)** //成功返回0，失败返回-1

对于普通文件，若文件长度大于length，额外的数据会被丢弃；若文件长度小于length，则扩展文件大小到length

对于Posix共享内存对象，ftruncate把该对象的大小设置成length字节

我们调用ftruncate来指定新创建的Posix共享内存对象大小，或者修改已存在的Posix共享内存对象大小。

* **mmap()函数**

mmap函数把一个文件或一个Posix共享内存对象映射到调用进程的地址空间，使

用该函数有三个目的：

* 使用普通文件以提供内存映射IO
* 使用特殊文件以提供匿名内存映射
* 使用Posix共享内存对象以提供Posix共享内存区

//成功返回映射内存的起始地址，失败返回MAP\_FAILED

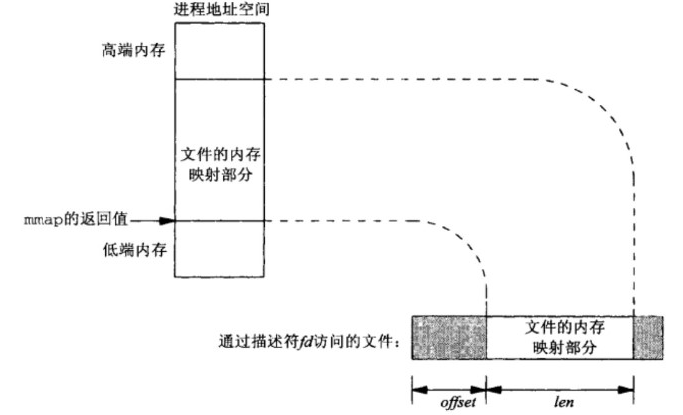
void \*mmap(void \*addr, size\_t len, int prot, int flags, int fd, off\_t offset);

mmap参数解析：

* addr指定映射内存的起始地址，通常设为NULL，让内核自己决定起始地址
* len是被映射到调用进程地址空间中的字节数，它从被映射文件fd开头起第offset

个字节处开始算，offset通常设为0，下图展示了这个映射关系

* prot指定对映射内存区的保护，通常设为PROT\_READ | PROT\_WRITE
* flags必须在MAP\_SHARED和MAP\_PRIVATE这两个标志中选择指定一个，进程间共享内存需要使用MAP\_SHARED
* 可移植的代码应把addr设为NULL，并且flags不指定MAP\_FIXED



mmap成功返回后，可以关闭fd，这对已建立的映射关系没有影响。

* 子进程设计

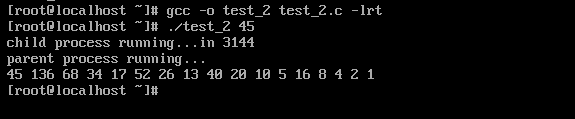
由于无法确实输出序列的长度，因此共享内存需要在子进程中重新计算序列所占的

内存大小，并调用ftruncate函数修改已存在的共享内存对象的大小

三、编码实现



四、运行截图



五、运行结果分析

子进程计算将序列存入共享内存，父进程等待子进程结束后，访问共享内存内容输出该序列

**实验内容三：普通管道通信**

一、需求分析和描述

设计一个程序，通过普通管道进行通信，让一个进程发送一个字符串消息给第二个进程，第二个进程收到此消息后，变更字母的大小写，然后再发送给第一个进程，比如，第一个进程发消息：”I am Here”，第二个进程收到后，将它改变为：“i AM hERE”之后，再发给第一个进程

(1)需要创建子进程，父子进程之间通过普通管道进行通信

(2)需要建立两个普通管道

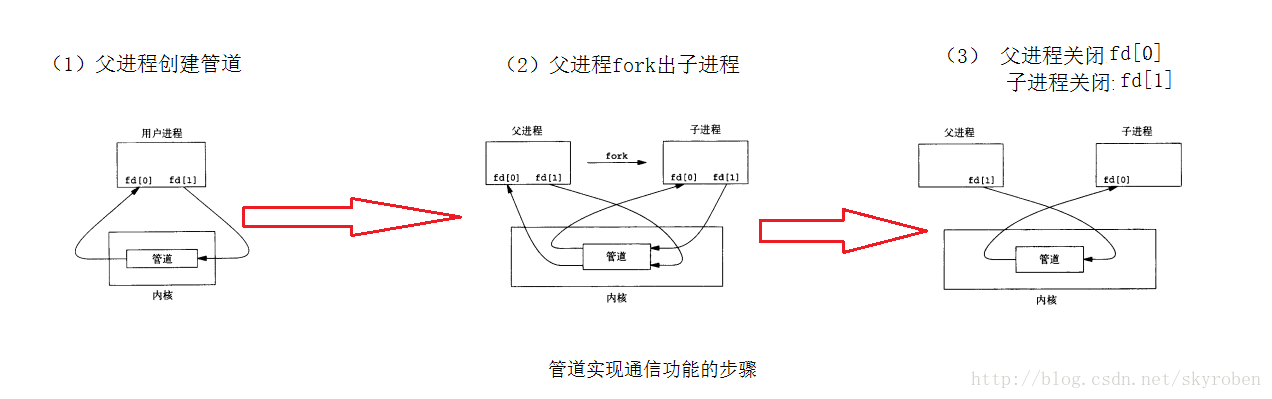
二、实验设计

普通管道通信的方式为：

（1）父进程创建管道，得到两个⽂件描述符指向管道的两端

（2）父进程fork出子进程，⼦进程也有两个⽂件描述符指向同⼀管道。

（3）父进程关闭fd[0],子进程关闭fd[1]，即⽗进程关闭管道读端,⼦进程关闭管道写端（因为管道只支持单向通信）。⽗进程可以往管道⾥写,⼦进程可以从管道⾥读,管道是⽤环形队列实现的,数据从写端流⼊从读端流出,这样就实现了进程间通信。

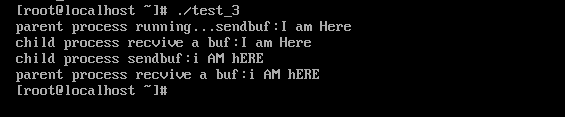


本题中首先关闭父进程的读端口，将字符串内容通过管道从父进程的写端口发送到子进程的读进程端，子进程从读端口收到信息后，处理信息，然后关闭父进程的写端口，关闭子进程的读端口，通过子进程的写端口将信息通过管道传到父进程的读端口，并关闭子进程写端口，之后父进程关闭读端口

三、编码实现



四、运行截图



五、运行结果分析

父进程通过管道发送内容”I am Here”给子进程，子进程收到信息后处理信息将大小写互换后，再通过子进程建立的管道将内容”i AM hERE”发送给父进程