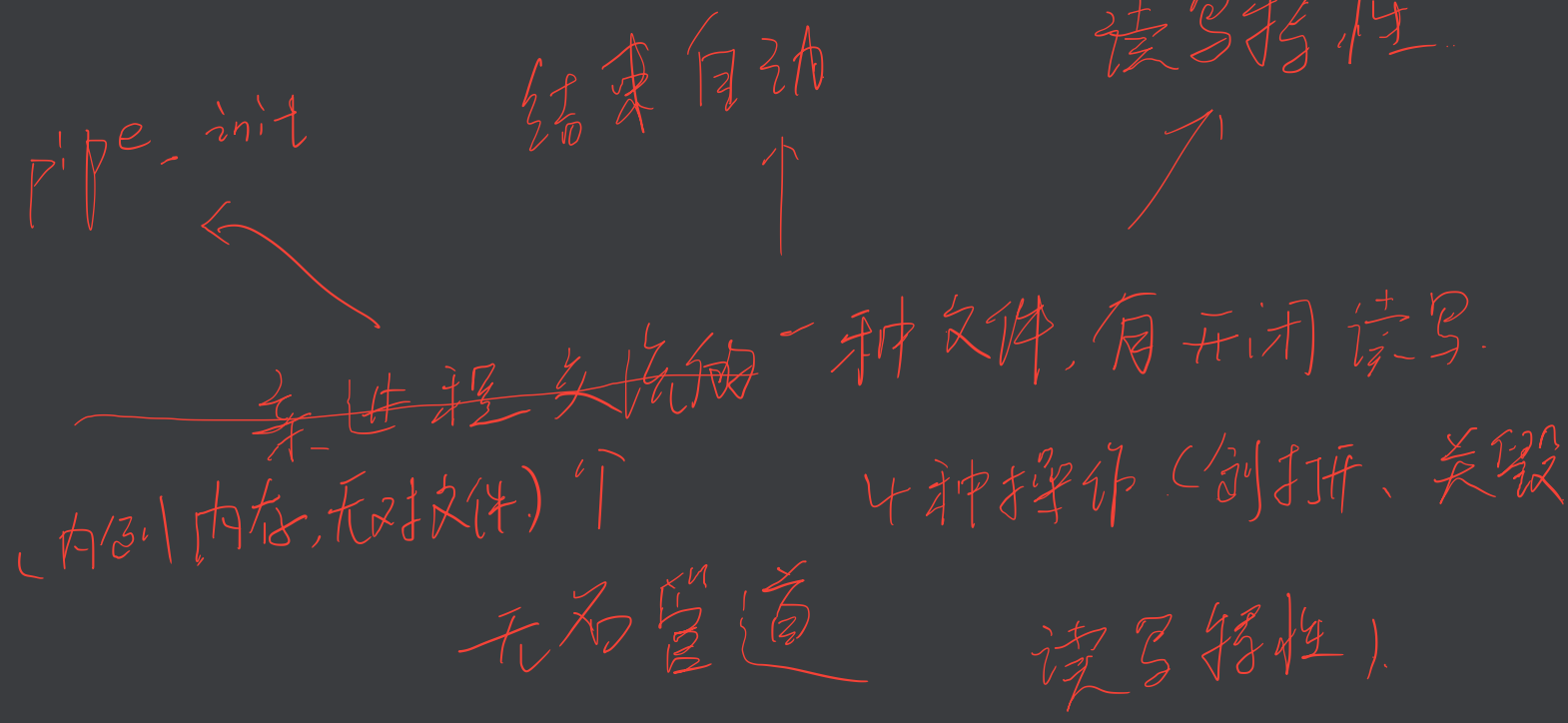
进程间通信（一）

主讲：大海老师

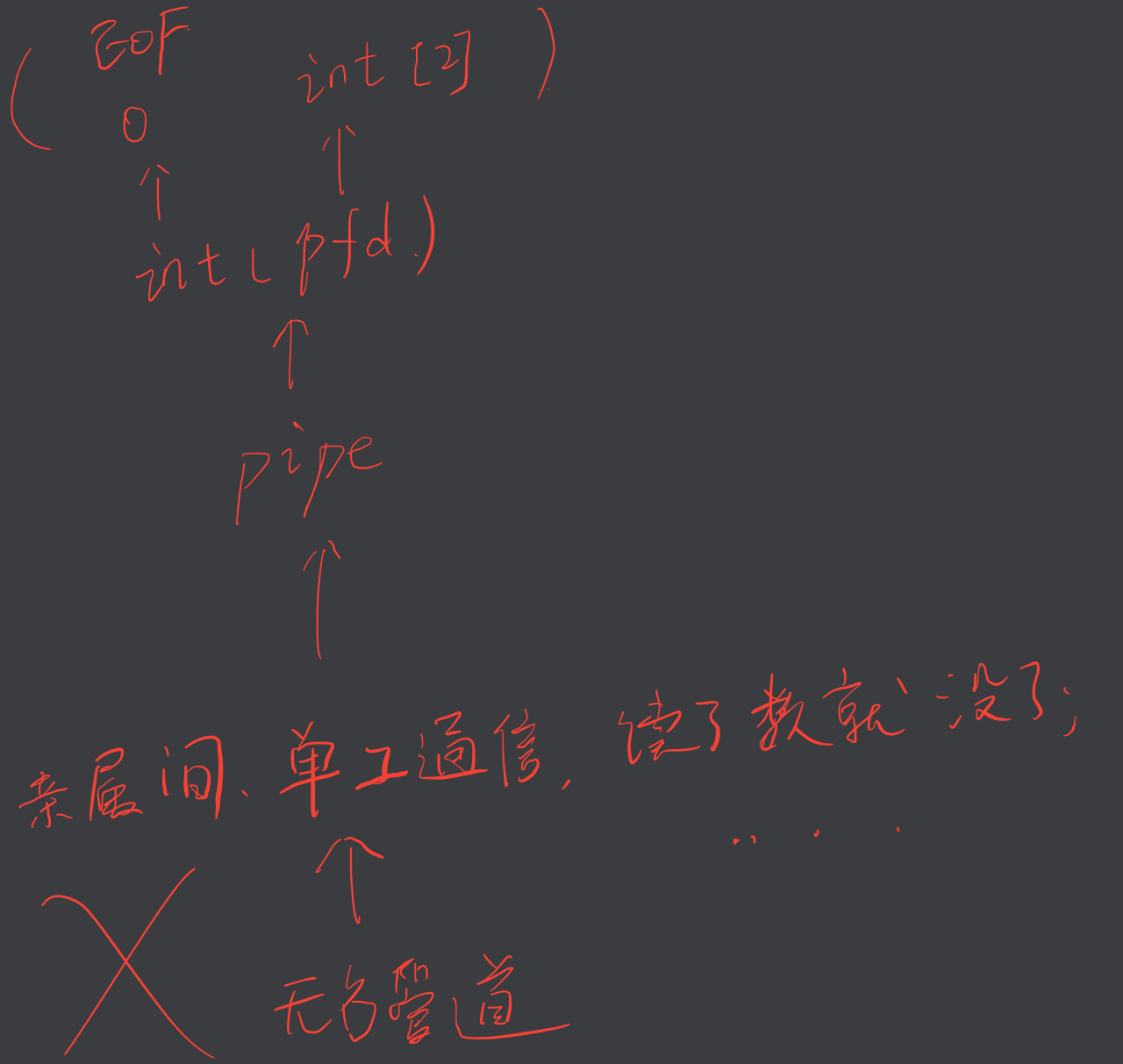
课程目标：

- 进程间通信方式介绍（了解）
- 无名管道特点（理解）
- 无名管道创建（熟练）
- 小结



进程间通信介绍

- 无名管道（pipe）
- 有名管道 （fifo）
- 信号（signal）
- 共享内存(mmap)
- 套接字（socket）



进程间通信介绍

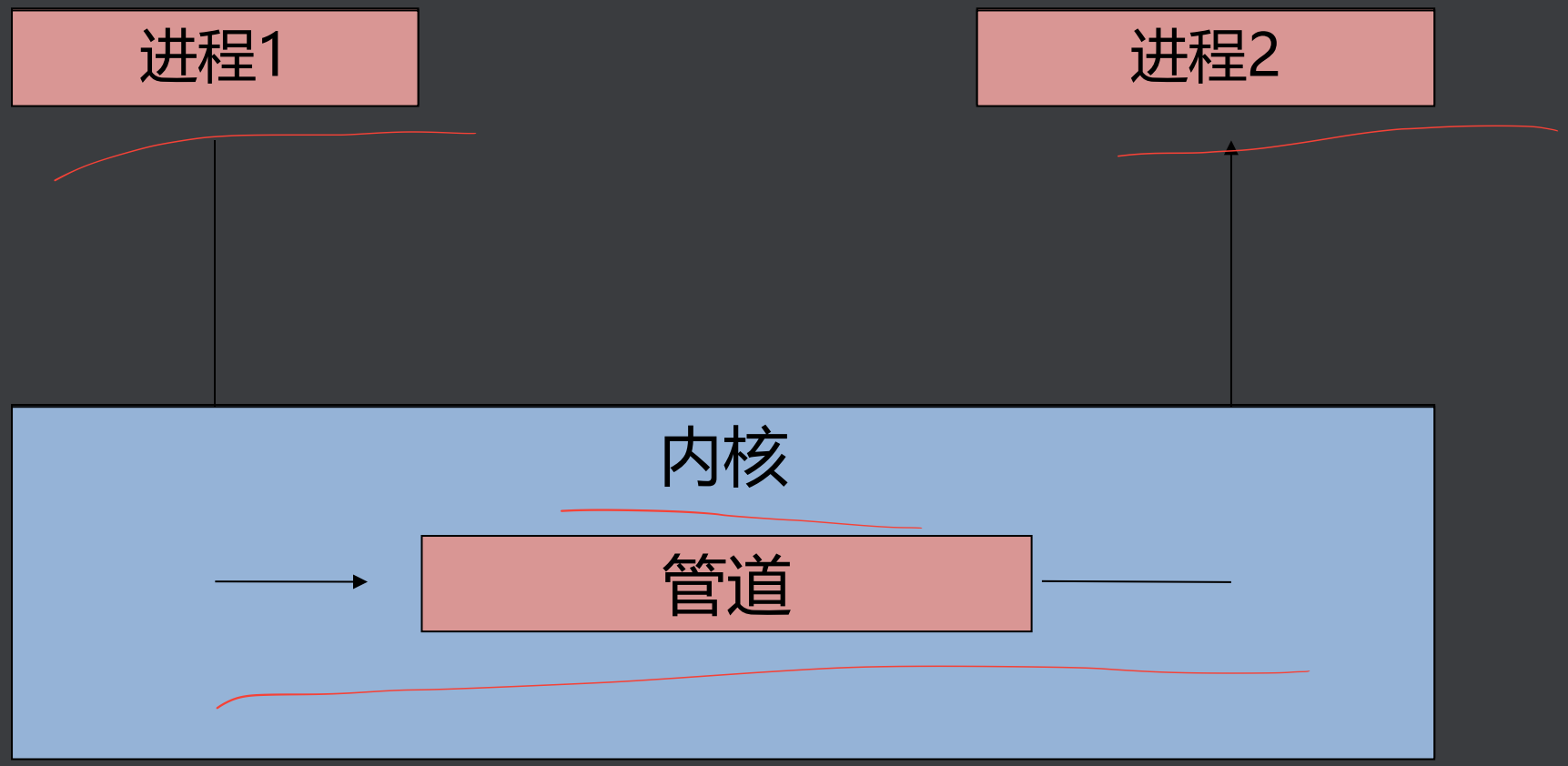
System V IPC

共享内存（share memory）

消息队列（message queue）

信号灯集（semaphore set）

无名管道



无名管道特点

无名管道具有如下特点：

只能用于具有亲缘关系的进程之间的通信

单工的通信模式，具有固定的读端和写端

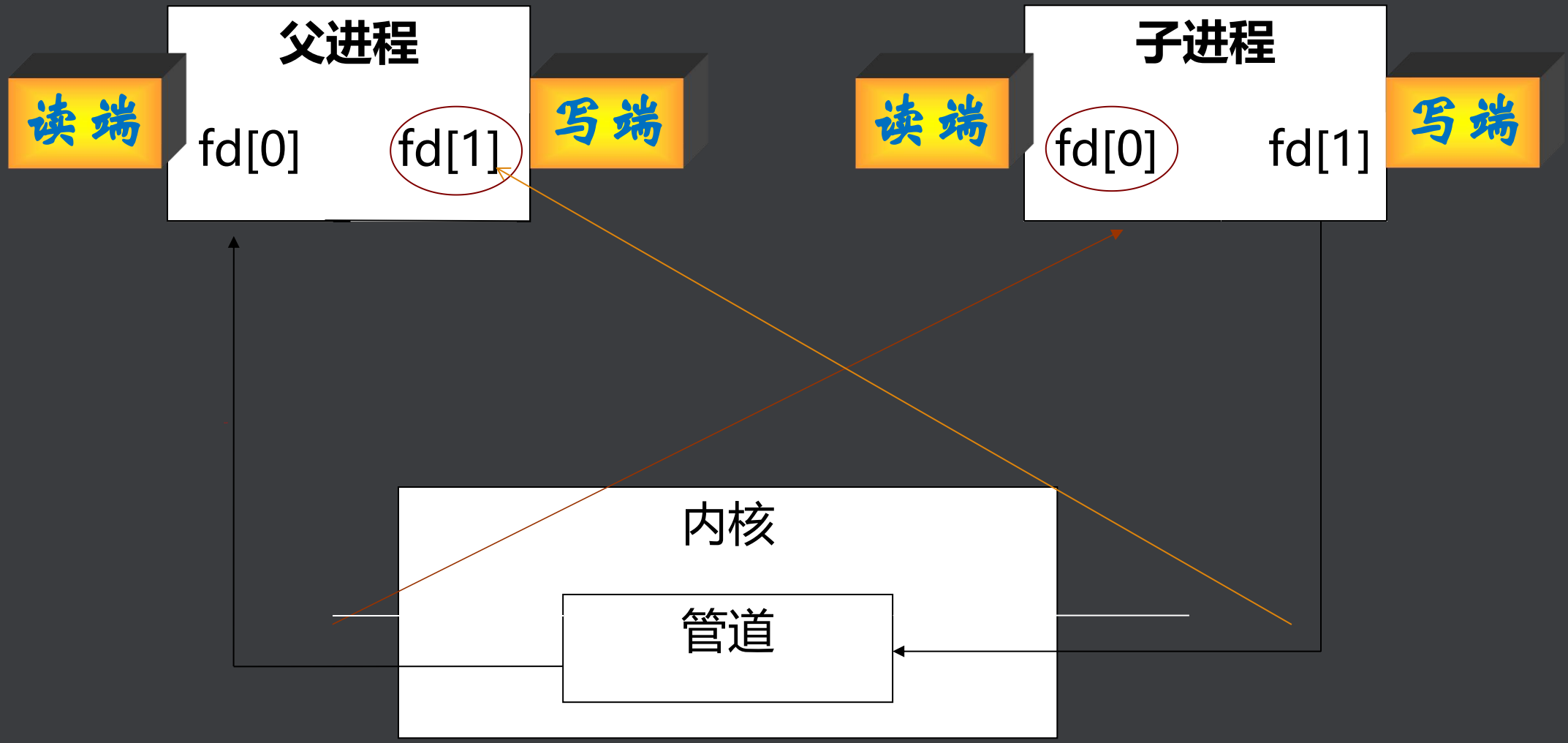
无名管道创建时会返回两个文件描述符，
分别用于读写管道

无名管道创建 – pipe

```
#include  <unistd.h>
int  pipe(int pfd[2]);
```

- 成功时返回0，失败时返回EOF
- pfd 包含两个元素的整形数组，用来保存文件描述符
- pfd[0]用于读管道；pfd[1]用于写管道

无名管道通信



无名管道 – 示例

子进程1和子进程2分别往管道中写入字符串；父进程读管道内容并打印；

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

int main(void) {
    pid_t pid1, pid2;
    char buf[32];
    int pfd[2];
    if (pipe(pfd) < 0) {
        perror( “pipe” ); exit(-1);
    }
```

pipe
fork
write pfd[1]
fork
write pfd[1]
wait
read

无名管道 – 示例

```
if ((pid1 = fork()) < 0) {
    perror( “fork” );  exit(-1);
}
else if (pid1 == 0) {          //  子进程1
    strcpy(buf,  “I’ m process1” );
    write(pfd[1], buf, 32);
    exit(0);
}
else {                 //  父进程
    if ((pid2 = fork()) < 0) {
        perror( “fork” ); exit(-1);
    }
    else if (pid2 == 0) {        //  子进程2
        sleep(1);
```

wait
read
Cstrcpy) (printf).

Q 都执行吗？觉得和之前学的

有点不一样呢？

都的话，前后顺序又是怎样呢？

A 从fork开始，有2进程从同一地点开始运行了。

Q 这么用也行吧

好像49.9多字节可原子操作

无名管道 – 示例

```
        strcpy(buf, “I’ m process 2”);
        write(pfd[1], buf, 32);
    }
    else {          // 父进程
        wait(NULL);
        read(pfd[0], buf, 32);
        printf(“%s\n”, buf);
        wait(NULL);
        read(pfd[0], buf, 32);
        printf(“%s\n”, buf);
    }
  }
  return 0;
}
```

无名管道小结

无名管道特性

pipe