进程概念：

系统编程：系统调用接口以及进程的认识（一个程序的运行）

进程概念***/进程控制**/基础IO**/进程间通信**^/进程信号**/多线程****

进程概念：

1、认识冯诺依曼体系结构

冯诺依曼体系结构：现代计算机硬件体系结构

输入设备：键盘

输出设备：显示器

存储器：内存条

运算器：中央处理器CPU

控制器：

硬件结构决定软件行为/ 所有设备都是围绕存储器工作的

2、简单认识操作系统

操作系统：内核+外部应用

操作系统的功能：管理计算机软硬件资源

操作系统的定位：一款“搞管理”的软件

目的：让计算机更加好用

如何管理：先描述，在组织

用户----->用户操作接口（shell命令、库函数）------>系统调用接口 ------>操作系统------->驱动程序------>底层硬件

库函数与系统调用接口的关系：库函数封装了系统调用接口；上下级的调用关系

3、进程概念

什么是进程：运行中的程序

进程：站在操作系统的角度，进程就是一个运行中程序的描述——PCB(进程控制块)【Linux下这个PCB实际上是一个结构体-struct task_struct{.....}】

进程如何描述一个运行中的程序：

内存指针：包括程序代码和进程相关数据的指针，还有和其他进程共享的内存块的指针

程序计数器：程序中即将被执行的下一条指令的地址

上下文数据：进程执行时处理器的寄存器中的数据(上一次正在处理的数据)
标识符：PID进程ID
进程状态：任务状态
优先级：让操作系统运行的更佳良好
记账信息：一个进程在CPU上运行的时间
I/O状态信息：
CPU的分时机制：每个程序在CPU上运行都有一个时间片
时间片：程序在CPU上运行的这段时间，运行完毕则调度切换
进程就是PCB

4、进程状态/优先级/环境变量/进程调度

进程状态：

如何查看进程：
查看进程命令：ps -ef/ps -aux（aux比ef更详细）
pid_t getpid(void)：获取调用进程的ID
pid_t fork(void):通过复制父进程创建一个子进程，子程序因为拷贝了父进程pcd里边的很多数据因此与父进程内存指针以及程序计数器都相同，所以运行的代码以及运行的位置都一样（父子进程运行的代码是一样的，并且子进程也是从创建成功的下一句指令开始运行）

返回值有三种：
-1：创建子程序失败
==0：对于子进程，返回值为0
>0：对于父进程，返回值是子进程的pid

fork创建子进程之后，**父子进程代码共享，数据独有**；
返回值：fork的返回值对于父子进程是不一样的
在父进程中，fork会返回子进程的pid
在子进程中，fork会返回0

可以通过返回值对父子进程的运行进行分流，让其各自运行一段代码
进程状态：运行　就绪　阻塞

Linux下的进程状态：

运行状态：R

可中断休眠状态：S

不可中断休眠状态：D(磁盘休眠状态)

停止状态：T

死亡状态：X

追踪状态：t

僵死状态：Z

kill 杀死一个进程（kill -9 强杀）

僵尸进程：处于僵死状态的进程

子进程先于父进程退出，若父进程没有关注子进程退出在子程序会进入僵死状态称为僵尸进程。子进程退出之后并没有完全释放资源，保存退出返回值（返回原因），操作系统检测到之后，会通知父进程，但是当前父进程没有关注到这个通知，因此操作系统无法直接释放子进程所有资源，子进程则进入僵死状态称为僵尸进程。

产生原因：子进程先于父进程退出，父进程没有关注到子进程退出状态，因此子进程成为僵尸进程，子进程退出，保存退出原因，操作系统不能主动释放，只能让父进程决定

僵尸进程的危害：资源泄露

如何处理：退出父进程

如何避免僵尸进程的产生：进程等待

nums/zombie/orphan/g　：将弟num行内容中的zombie字符串替换成orphan字 符串

num1，num2s/zombie/orphan/g:从num1行到num2行进行内容替换

孤儿进程：

父进程先于子进程退出，则子进程会成为孤儿进程，这个孤儿进程运行在后台，并且被1号init进程领养。

守护进程/精灵进程：是一个特殊的孤儿进程，完全独立的运行系统后台，脱离终端的影响----------课后可以调研一下

进程优先级：决定进程cpu资源的优先分配权的等级

优先级：决定一个进程在CPU上运行的优先权----数字

数字越小，优先级越高

程序分类：CPU密集/IO密集

进程分类：批处理/交互式

renice -n 19 -p pid ： 设置进程优先级

sudo nice -n -20 ./zombie

环境变量：保存系统运行环境参数的变量

作用：通过环境变量配置系统运行环境参数可以使系统环境配置起来更加灵活（修改环境变量，则配置立即生效）,可以通过环境变量给一个进程传递参数

操作命令：

env：查看所有环境变量

set：查看所有变量

echo：打印指定变量的值

export：声明一个环境变量

unset：删除一个环境变量

目的：

1、让系统运行环境参数配置起来更加灵活

2、环境变量具有全局特性

典型的环境变量：

PATH:程序运行的默认搜索路径

SHELL/USER/PWD/HOME

int main(int argc,char *argv[],char *env[]):argc程序运行参数的个数，argv保存程序运行参数，env保存环境变量

extern char **environ；使用全局变量

char * getenv(const char *name)：通过指定的名称获取环境变量的值

调研：putenv()/setenv()

创建一个进程，子进程可以获取到所有的环境变量

5、程序地址空间

内存地址：内存区域的编号

进程地址空间：

在程序中看到的所有变量的地址其实都是一个虚拟地址

程序地址空间，是一个虚拟地址空间

通过页表映射物理内存区域：避免了程序直接操作物理内存

如何将虚拟地址映射转换为物理地址：

分段式内存管理：对程序编译时内存管理更加友好

内存地址：段号+段内偏移;->段表(物理内存段的起始地址)->物理地址

分页式内存管理：主要用于提高内存利用率

内存地址：页号+页内偏移;->页表(物理块号)->物理地址

段页式内存管理：在将程序地址空间进行分段式管理，但是在每个段内进行分页式管理

内存地址：段号+段内页号+页内偏移->段表(段内页表地址)/段内页表

(物理块号)->物理地址

虚拟地址空间：操作系统为进程描述的一个连续的、完整的地址空间---mm_struct{ulong size,ulong star;ulong end}，每个进程都有自己独立的虚拟地址空间，访问的都是虚拟地址，但是可以通过页表将虚拟地址映射转换为物理内存地址，进而访问物理内存区域；通过页表映射转换实现了数据在物理内存中的离散式存储------提高了内存利用率；通过页表中进行地址访问限制，实现了内存访问控制，并且每个进程都有独立的虚拟地址空间，访问的都是自己的虚拟地址，因此进程之间具有独立性。

作用：

提高内存利用率：物理内存的离散存储

保证了进程独立性：每个进程都只能访问自己虚拟地址映射的物理内存

页表可以进行内存访问控制：页表可以对每个虚拟地址进行权限标记

物理地址的计算方法：

虚拟地址/页面大小=页号 通过页表得到块号

块号*块大小+页内偏移（虚拟地址%页面大小）

创建一个子程序的流程：写时拷贝技术

1、创建pcd

2、拷贝父进程pcd中的数据（拥有相同的虚拟地址空间，相同的页表。。）

3、父子进程一开始映射同一块物理内存

4、等到物理内存修改的时候才为子进程重新开辟内存，拷贝数据过来（要求i进程的独立性）

代码共享和数据独有：因为代码段只是只读的，不会修改，因此会一直映射同一块物理内存

进程调度算法：大O(1)调度算法(多级反馈)