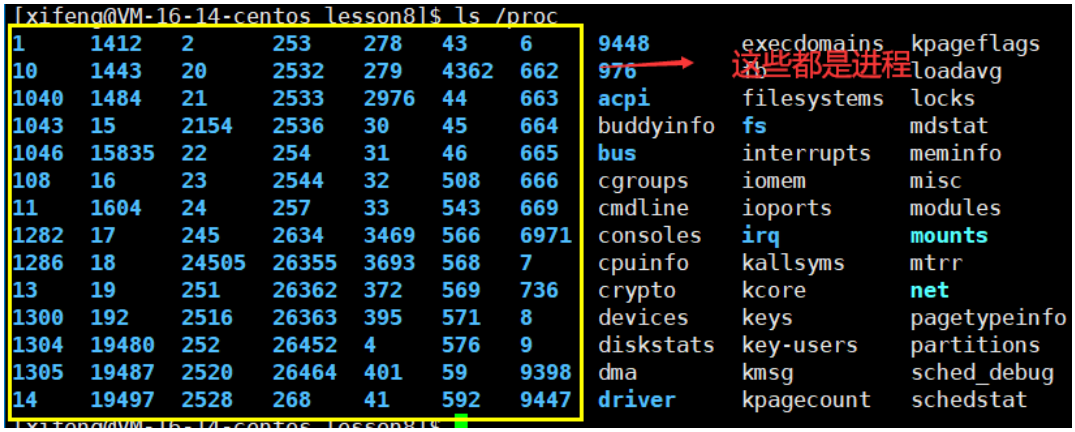
进程

补vim上面的一些操作:

- 命令模式下**ctrl+v**,然后通过**hjkl**来选中区域，然后**输入A输入//最后按esc** 来进行批量化注释
- 命令模式下**ctrl+v**,然后通过**hjkl**来选中区域，然后**按d**就可以批量化删除了

补查看进程的方式:

- ls -d /proc(查看pid为1286的进程ls -al /proc/1286)
  - 

  - 当进程开始的时候就会在/proc 目录里面创建一个目录，进程结束后会自动消失
  - 

  - cwd 就是当前工作目录

如何管理进程

先描述，在组织(描述进程的结构体--PCB进程控制块)

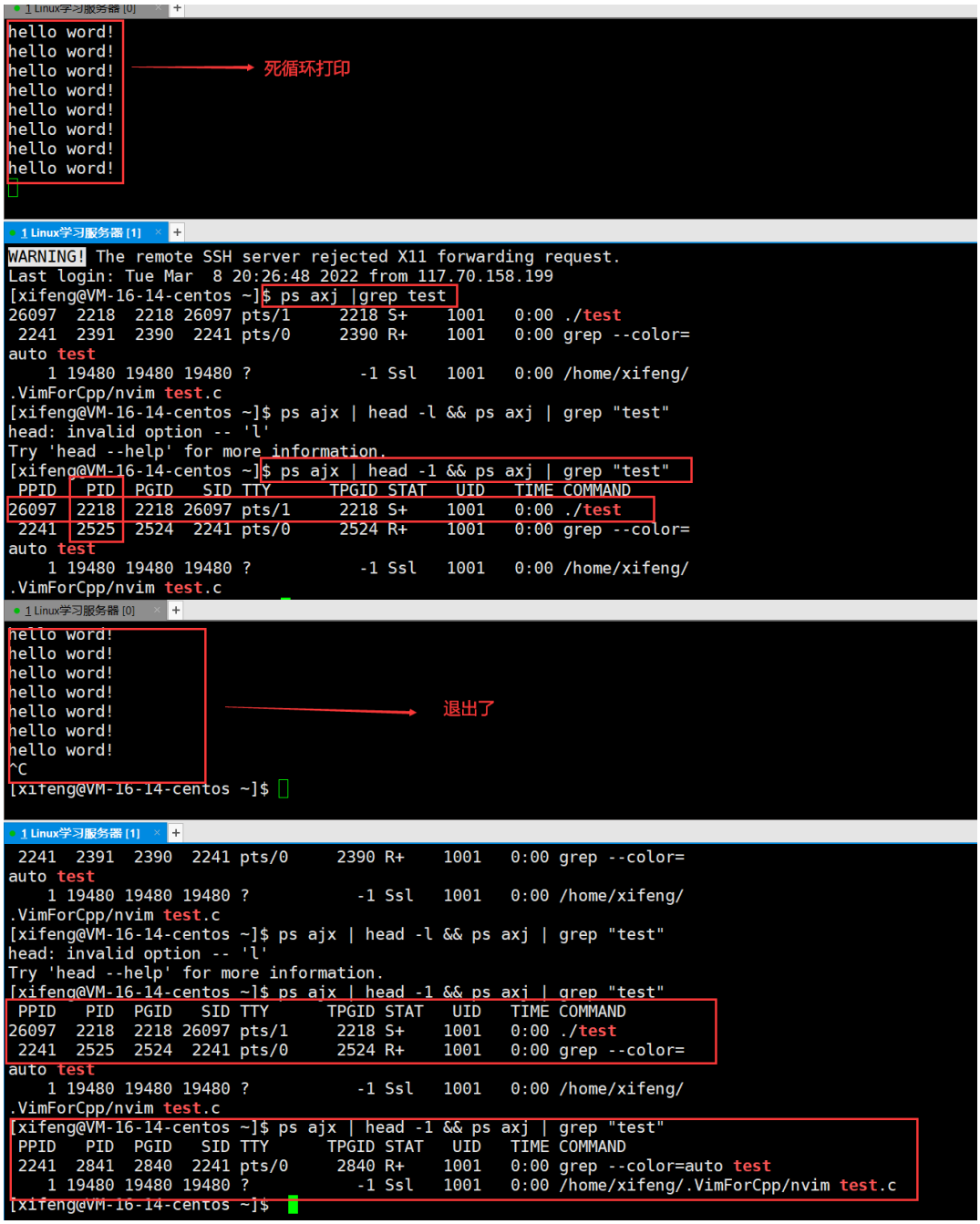
操作系统不信任任何用户---->给用户提供各种**系统调用接口**(接口实际上就类似于C的库函数)

进程大致了解:

- 加载到内存的程序，就叫做进程
- 任何进程在形成时，操作系统要为该进程创建**PCB，进程控制块**

  在linux系统中，PCB---->struct task_struct{}
  - ```
    //简单理解就是
    struct task_struct{
        //进程的所有的属性数据
    }
    ```
    ```

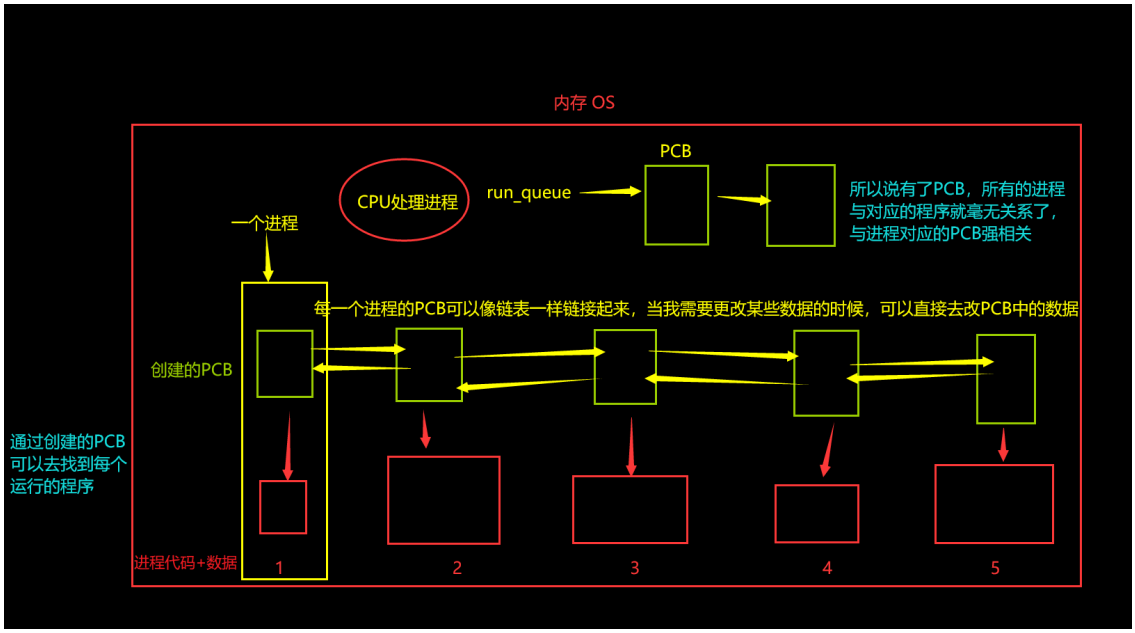
- 我们启动程序的过程本质都是在系统上面创建进程！！(查看进程：**ps axj |grep "test"**)(详细：**ps axj |head -1 && ps axj |grep "test"**)



进程 vs 程序

进程=程序 +操作系统维护进程相关的数据结构

- 有了进程控制块，所有的进程管理任务与进程对应的程序毫无关系！！与进程对应的内核(OS)创建的该进程的PCB强相关！！



PCB的内部构成

- **标识符**：描述本进程的唯一标识符(**PID**)，用来区别其他进程
  - **getpid 获取进程自身的id**

```
GETPID(2)                  Linux Programmer's Manual

NAME
       getpid, getppid - get process identification

SYNOPSIS
       #include <sys/types.h>
       #include <unistd.h>

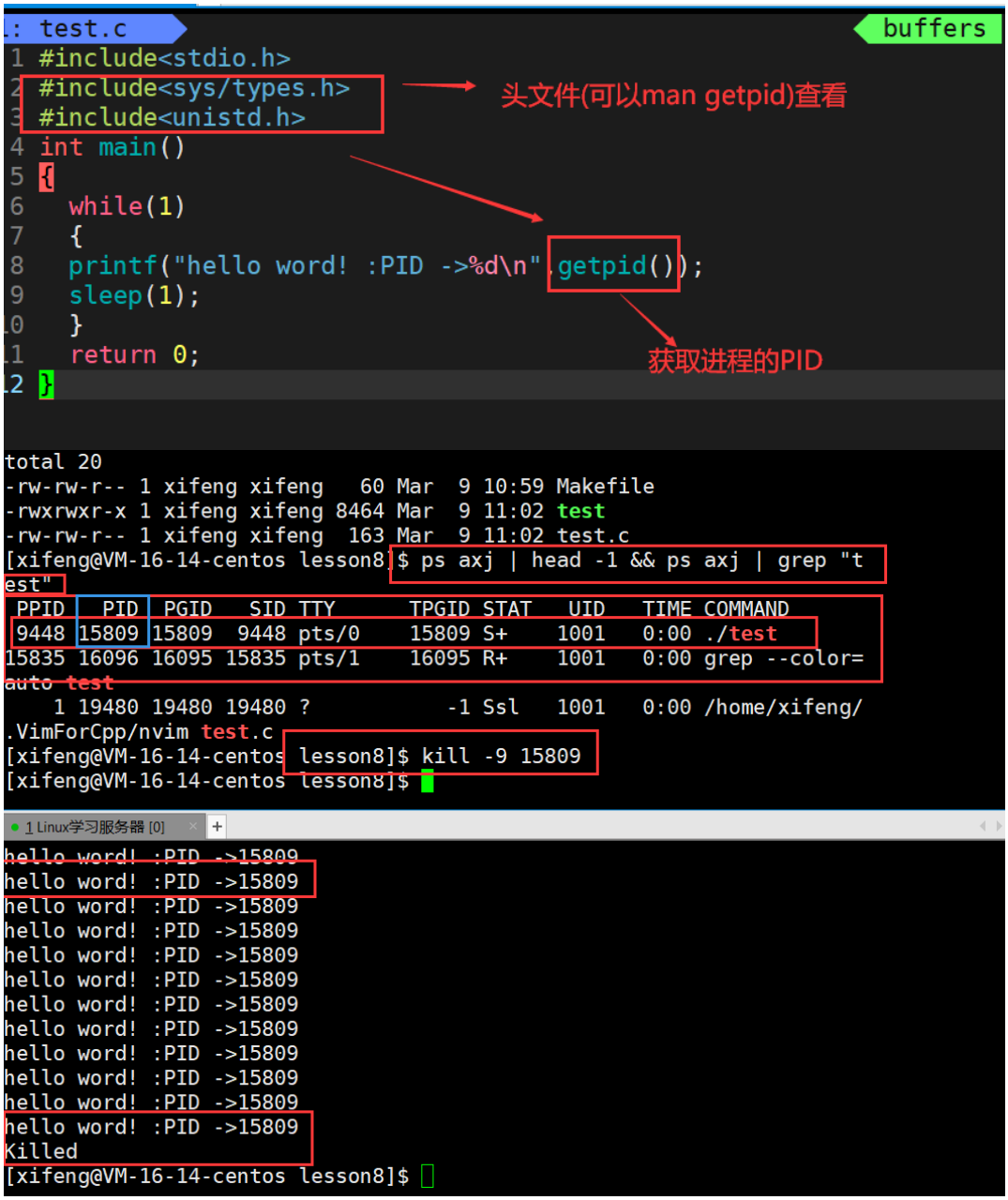
       pid_t getpid(void);
       pid_t getppid(void);
```

  - **getppid获取父进程id**(在命令行上运行的命令，基本上父进程都是bash)

```
[xifeng@VM-16-14-centos lesson8]$ ps axj | grep 9448
 9447  9448  9448  9448 pts/0    17288 Ss    1001   0:00 -bash
 9448 17288 17288  9448 pts/0    17288 S+    1001   0:00 ./test
15835 18320 18319 15835 pts/1    18319 R+    1001   0:00 grep --color=auto 9448
[xifeng@VM-16-14-centos lesson8]$
```

```
hello word! :PID ->17288  PPID->9448
hello word! :PID ->17288  PPID->9448
hello word! :PID ->17288  PPID->9448
```

  - **echo $?** 输出最近执行命令的**退出码**
  - **kill -9 PID**就可以结束进程



- **状态**
  - 任务状态，退出代码，退出信号等等
- **优先级**：相对于其他进程的优先级
  - 先后问题 vs 权限

    权限决定的是能不能

    优先级是已经可以只是顺序问题
- **程序计数器**：永远指向当前程序正在执行指令的下一条指令的地址
- **内存指针**(可以简单理解为通过内存指针可以联系到进程的实体)：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
- **上下文数据：进程执行时，所形成的处理器寄存器当中的和进程强相关的临时数据**
  - 理解上下文

    1. 操作系统规定每个进程单次运行的**时间片**(就是一个进程单次在CPU上运行的最长时间)
    2. 在一个CPU情况下，用户感受到的多个进程同时在运行，本质是通过CPU的快速切换完成的

    进程在运行期间是有切换的，进程可能存在大量的临时数据--->暂时在CPU的寄存器中保存

        - **保护上下文和恢复上下文**：为了让我们去做其他事情，但是不耽误当前学习，并且，当我们回来继续的时候，可以接着之前学习的内容继续学习
    - **通过上下文，我们就能感受到进程是被切换的**
- **I/O 状态信息**：包括I/O请求，分配给进程的I/O设备和被进程使用的文件列表
    - OS 调度模块，较为均衡的调度(调度：获得CPU资源)每个进程
- **记账信息：可能包括处理器时间总和，使用的时钟数总和，时间限制，记账号等**

进程=程序文件内容 +相关的数据结构