块设备、进程和作业管理

主要内容

- 块设备管理
- 进程和作业管理

dd命令 dd [option]

- 以block为单位从标准输入或文件中读取数据，输出到标准输出或者文件中
  - if=IFILE : 从文件中读，如果IFILE为块设备时以“原始形式"读
  - of=OFILE：写入到文件中，OFILE为块设备时以”原始形式“写
  - bs=BYTES: 块大小为BYTES，即读写时一次读写BYTES个字节，缺省512字节
  - count=N：总共读写N块，实际读写的数据为 count*bs，如果不指定，则读完为止

BYTES 后面还可添加单位，如b(512字节), kB(1000字节), K(1024字节)， MB=10^6字节，M为2^20字节等

dd if=/dev/sdx of=/dev/sdy	将整个硬盘sdx备份到另一个硬盘sdy，潜在磁盘杀手，当心！！！
dd if=/dev/sr0 of=cd.iso	光盘中的内容制作成ISO映像
dd if=/dev/sda2 of=/path/to/image	将分区sda2备份到映像文件
dd if=/path/to/image of=/dev/sda2	将备份映像恢复到分区sda2
dd if=/dev/urandom of=/dev/sdc1	利用随机数据填充硬盘，销毁数据，不要随便尝试!

```
$ dd if=/dev/zero of=file.img bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.140799 s, 745 MB/s
$ ls -l file.img
-rw-r--r-- 1 dlmao dlmao 104857600 Apr 28 23:44 file.img
```

创建文件系统:mkfs

- lsblk 可以查看当前所有块设备
- cat /proc/filesystems 可以看到当前主机所支持的文件系统
- **mkfs –t type device**: 通用的命令，在device对应的块设备上建立一个类型为type的文件系统。
- mkfs.ext4 (ext2/ext3) device创建Linux文件系统，相当于mkfs -t ext4 device
- mkfs.vfat device创建Fat文件系统，相当于mkfs -t vfat device
- mkfs.ntfs device创建NTFS文件系统，相当于mkfs -t ntfs device

```
$ mkfs -t ext4 file.img
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 102400 1k blocks and 25688 inodes
Filesystem UUID: 664cd116-3fdd-4674-b139-29d1a6c2caeb
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
$ mkdir mnt; sudo mount file.img mnt
$ sudo chown -R $(id -un): mnt
```

```
$ lsblk | grep -v loop
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  1.1G  0 disk
sdb      8:16   0   20G  0 disk
├─sdb1   8:17   0   18G  0 part /
├─sdb2   8:18   0    1K  0 part
└─sdb5   8:21   0    2G  0 part [SWAP]
sr0     11:0    1 56.5M  0 rom
sr1     11:1    1   16M  0 rom
```

挂载文件系统mount

```
dlmao@mars:~$ touch mnt/$(date +%m-%d).txt
dlmao@mars:~$ ls mnt
04-29.txt
dlmao@mars:~$ sudo mount file.img mnt
dlmao@mars:~$ ls mnt
lost+found
dlmao@mars:~$ sudo umount mnt
dlmao@mars:~$ ls mnt
04-29.txt
```

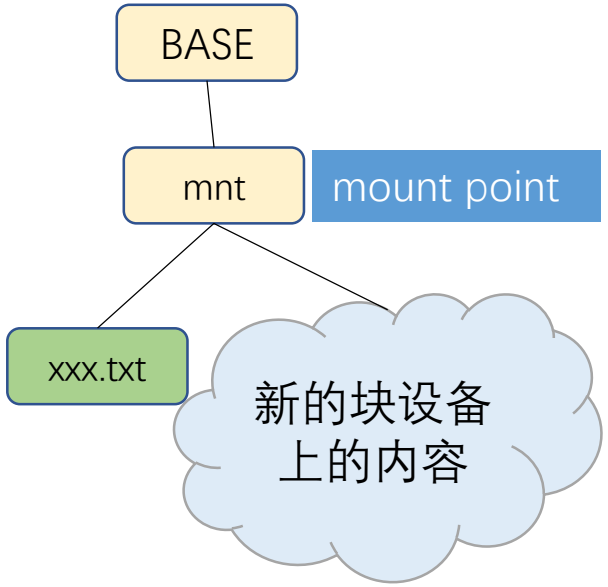
除非在/etc/fstab里面特别设置，挂载和卸载都要超级用户权限

mount 查看已经挂载的文件系统，findmnt

mount [-t fstype] [-o options] device dir

mount –a 自动挂载/etc/fstab给出的标记为auto的分区

mount device|dir 挂载/etc/fstab中设备或者目录所对应的文件系统



把device(对应某个块设备或块设备上的分区）挂载到某个目录dir

- 该设备的文件系统的内容可以通过挂载点访问
- **原挂载点下的内容暂时屏蔽不可访问**，等待umount后可以继续访问
- -t 选项可以指定挂载分区的文件系统类型(ext2, ext3, ext4, msdos,vfat,ntfs, iso9660等)，不指定时自动探测文件系统类型
- -o options：给出了挂载文件系统时使用的选项

umount [-l] directory|device 卸载已经挂载的文件系统，

- -l, --lazy 表示在设备忙时也卸载，等待在那些导致设备忙的进程退出时释放相应的资源

在挂载前mnt目录下xxx.txt可以访问，但是挂载文件系统到mnt之后，原来mnt目录下的内容暂时不可访问

自动挂载文件系统

```
dlmao@mars:/$ lsblk -f | grep -v loop
NAME    FSTYPE    LABEL UUID                                 MOUNTPOINT
sda
└─sda1 ext4            5daa65a1-ede6-4185-b598-c90ea58fab85 /
sdb    ext2      home2 0b2677b6-b4d7-4bb9-b1fb-1b81e6ab80ef
```

/etc/fstab 保存了经常挂载的文件系统的信息

- 块设备文件，可以是设备文件名，也可以是LABEL=<label>或UUID=<uuid> 如果其中有空格或制表，代替以\040或\011
  - LABEL为文件系统的逻辑卷标：命令e2label/dosfslabel/ntfslabel查看或修改卷标
  - UUID为唯一标识该文件系统的字符串（小写）：blkid device或者lsblk –f [device]可查看
  - 由于设备增删时设备文件名可能会变动，建议采用LABEL或者UUID
- 挂载点：对于swap，该字段为None
- 文件系统类型：待挂载的文件系统的可能的类型，可以有多个类型，之间以逗号隔开
- 选项：挂载文件系统时使用的选项，多个选项时以逗号隔开
- dump字段： 备份命令dump时使用该字段，为0表示不备份，1表示备份
- pass字段： 启动时使用fsck命令检查文件系统时，使用该字段决定检查顺序，0表示不检验，1表示最早检验(根文件系统) ,2表示最晚检验(一般的文件系统)

```
# <file system>                          <mount point> <type>  <options>          <dump>  <pass>
UUID=5daa65a1-ede6-4185-b598-c90ea58fab85 /      ext4  errors=remount-ro      0       1
/swapfile                                 none     swap  sw                     0       0
```

挂载文件系统mount

- -o options：给出了挂载文件系统时使用的选项, defaults 表示使用所有选项的默认值（ rw, suid, dev, exec, auto, nouser, async. ）

```
sudo mount  device|dir  #挂载/etc/fstab中设备或目录对应的文件系统
sudo mount –a   # 自动挂载在/etc/fstab中配置好的块设备
sudu mount –o remount,ro /dev/sdb  #重新挂载，以只读方式
sudo mount -o loop  /tmp/disk.img /mnt
sudo mount -o loop=/dev/loop3  /tmp/disk.img /mnt # 也可指定loop设备
sudo mount /tmp/disk.img /mnt #自动采用loop选项挂载
```

选项	含义	选项	含义
ro	只读方式挂载	rw	读写方式挂载
dev/nodev	是否对文件系统上的设备文件进行解释	exec/noexec	是否允许运行系统上的二进制文件
suid/nosuid	是否允许二进制文件的suid/sgid起作用	async/sync	是否同步方式
user/nouser	是否允许普通用户挂载(如果挂载成功卸载只有该用户或超级用户允许)	owner/noowner group/nogroup	允许设备文件的拥有者mount 允许设备文件的用户组成员mount
users	允许所有用户挂载和卸载	auto/noauto	是否可用-a选项挂载
remount	重新挂载已挂载的文件系统	loop	挂载loop设备

- 回环设备文件/dev/loop...是一种伪设备，使得文件可以如同块设备一样被访问
- mount命令的loop选项会自动创建对应着映像文件的loop设备，并且挂载该映像文件中的文件系统
- mount命令中，如果设备为一个普通文件，则等价于 -o loop选项

块设备新增和管理

危险操作：在使用fdisk/mkfs等命令时，确认所操作的设备文件无误！！！

- Virtualbox虚拟机设置→存储→控制器:SATA，可添加新的虚拟硬盘
- 虚拟机要使用外部的USB 2.0/3.0设备
  - 去virtualbox官网下载并安装配套的**VirtualBox Extension Pack**
  - **在安装好后，在相应虚拟机的设置→USB驱动器中选择启动USB3.0或2.0控制器**
  - VirtualBox中设备→USB→选择对应的U盘，可捕获该U盘供当前虚拟机使用
- 块设备对应的设备文件是什么？ lsblk 查看, lsblk -f查看文件系统相关(类型,卷标,UUID等)
- 整个磁盘可划分为多个分区: fdisk /gparted命令或桌面环境中的磁盘应用gnome-disks
- 创建文件系统mkfs：可整个磁盘，也可分区后的各个磁盘分区
- 挂载文件系统：mount到指定挂载点，测试好后更新/etc/fstab，下次就可自动挂载了

```
dlmao@mars:~$ lsblk | grep -v loop
sda       8:0    0    20G  0 disk
└─sda1    8:1    0    20G  0 part /
sdb       8:16   0     1G  0 disk /home2
sdc       8:32   0 536.8M  0 disk
sr0      11:0    1  1024M  0 rom
```

挂载FAT和NTFS文件系统

微软使用的FAT和NTFS文件系统实现与Linux下的文件系统不一致

- 权限管理：挂载时模拟Linux的权限控制的相关字段：uid, gid以及访问模式
  - 访问模式通过umask描述。针对文件和目录可指定不同的权限fmask或dmask
  - vfat文件系统缺省为执行mount的当前进程的uid、gid和umask
  - ntfs文件系统，uid/gid缺省为root，umask为000，即访问模式为rwxrwxrwx
  - 具体缺省值要看每个发行版的具体情况
- 文件名编码，Linux缺省本地化设置都采用utf-8编码
  - windows有短文件名，中文windows系统采用cp936即GBK编码，也有长文件名，长文件名采用utf-16编码
  - 对于vfat类型文件系统：建议选项codepage=936,utf8
  - 对于ntfs文件系统：建议选项nls=utf8

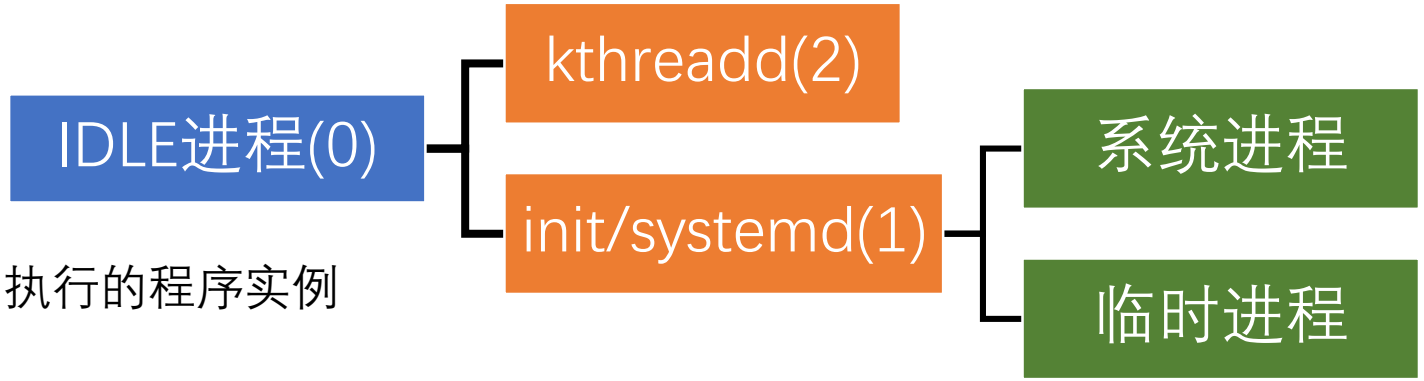
```
$ sudo mount /dev/sdc1 /mnt -o uid=$(id -un),gid=$(id -gn),dmask=022,fmask=133,utf8,codepage=936,iocharset=utf8
$ sudo mount /dev/sdc2 /media -o uid=$(id -un),gid=$(id -gn),dmask=022,fmask=133,nls=utf8
$ ls -ld /mnt /media
drwxr-xr-x 1 dlmao dlmao  4096 May  1 22:55 /media
drwxr-xr-x 3 dlmao dlmao 16384 Jan  1  1970 /mnt
```

- dmask=022 fmask=133　目录允许用户组和其他人读，文件允许用户组和其他人读，但是缺省不可执行
- uid和gid都设置为当前用户的uid和gid

主要内容

- 块设备管理
- 进程和作业管理

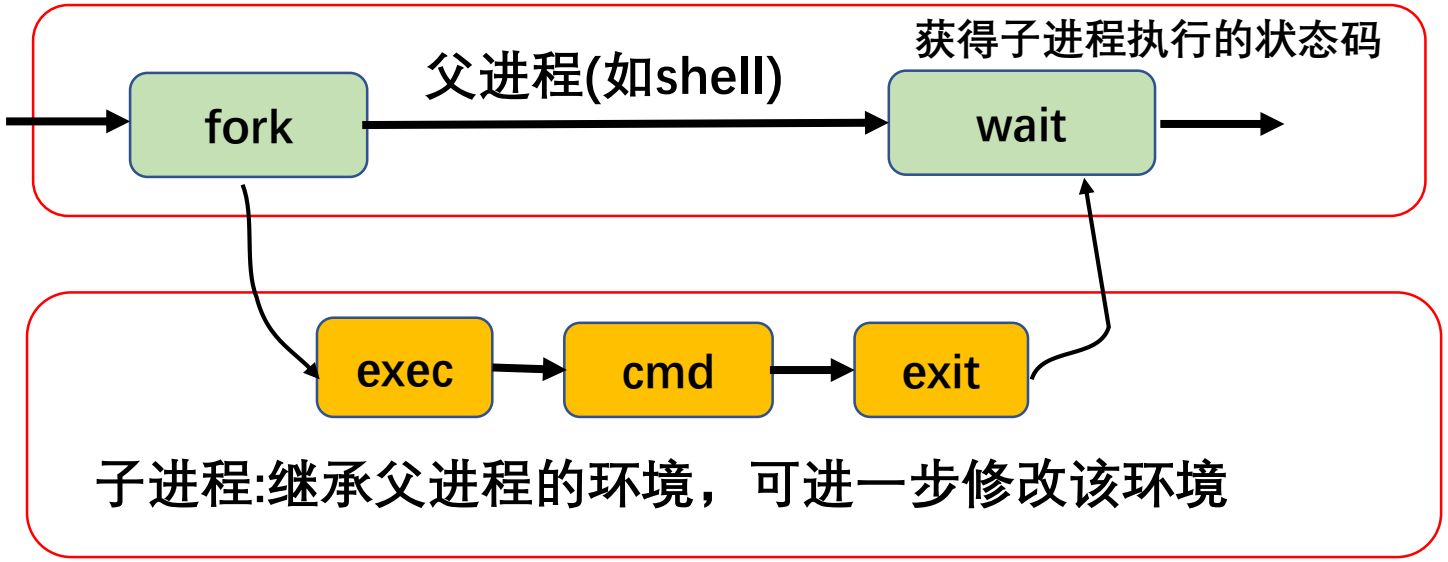
进程(Process)



- 程序(Program)：一个可以执行的文件
- 进程(Process)：指的是一个加载到内存中执行的程序实例
- 内核负责管理进程，维护了一个进程表
  - 每个进程分配一个唯一的ID，称为PID（进程ID）
  - 每个进程维护了一些状态信息
    - 用于权限控制的实际和有效用户ID、实际的用户组ID和有效用户组ID、umask
    - 工作目录、环境变量
    - 进程管理相关的进程ID、父进程ID、进程组ID、会话ID和控制终端等
  - 调度器进行时间片轮转调度，选择正在执行的多个进程中的其中一个或者多个，让其运行一段短的时刻(10毫秒CPU时间)
- 每个进程都有一个父进程。父进程启动一个子进程，子进程可再启动另一个进程，形成一棵进程树
- 系统引导时内核手动创建一个特殊的pid为0的空闲进程，然后由IDLE进程创建1号用户进程和2号内核进程
- 内核进程：协调各种资源访问，使用ps查看时，进程名以中括号包括，比如[kthreadd]
- 用户进程：位于用户空间的进程，包括长时间运行的系统进程或守护进程(daemon)和临时进程

父进程和子进程

- 父进程应该调用wait等待子进程结束，了解子进程执行的结果（状态码）
- 父进程退出时，其子进程由init进程收养，设置子进程的父进程ID(ppid)为1
- 已经结束但尚在进程表的进程称为**僵尸(ZOMBIE)进程**，保留一些必要的信息，以等待父进程获取子进程执行返回的状态码
- 父进程存在，但是不调用wait回收已结束的子进程，该僵尸进程被遗弃，通过kill杀死父进程，由init收养并回收该僵尸进程

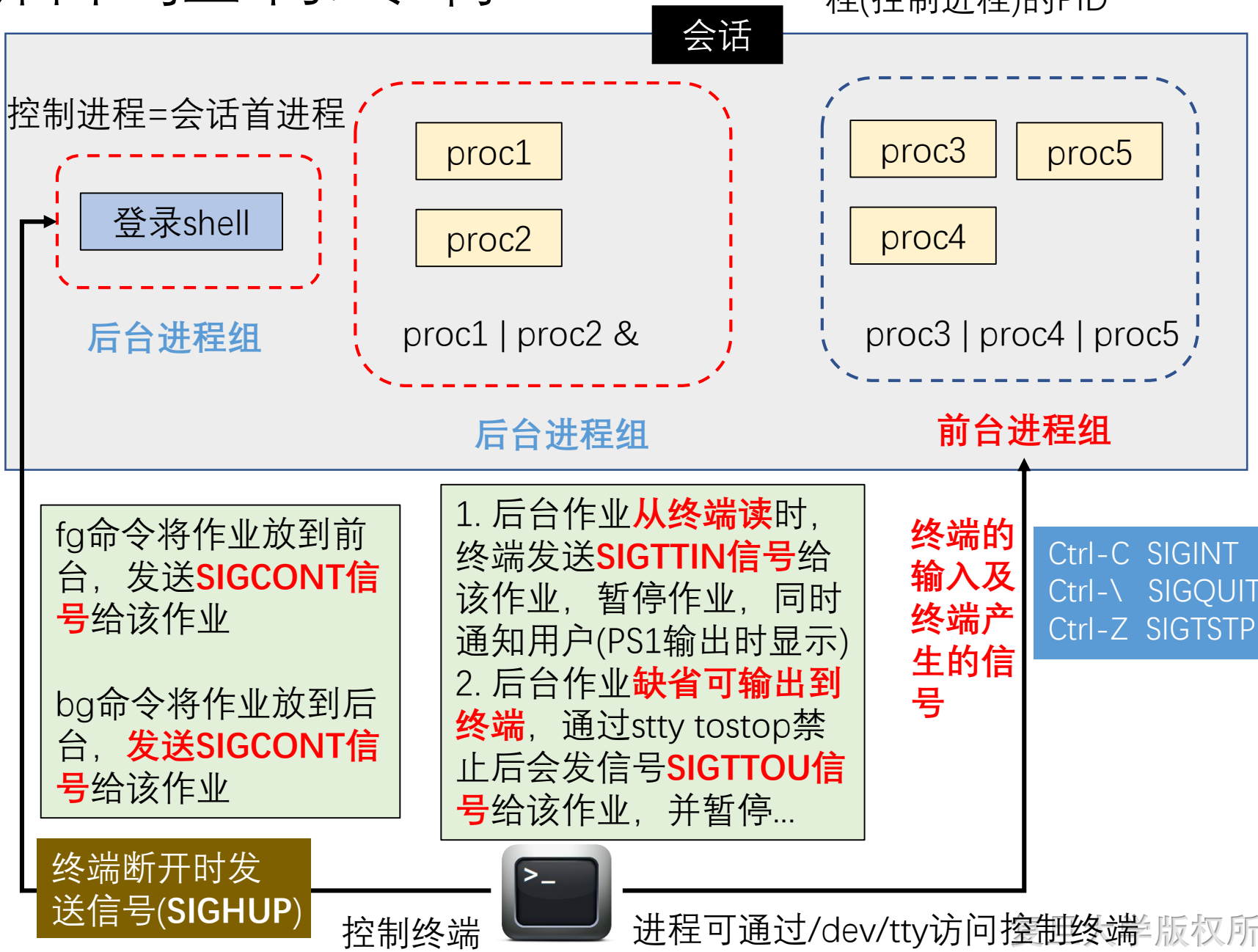


进程组、会话和控制终端

- 进程组是一个或多个进程的集合
  - 每个进程属于且只属于一个进程组，进程组的ID为该进程组组长的PID
  - 引入进程组的目的是**方便给进程组的多个进程发送信号**
  - 通过shell执行的外部程序对应着一个进程组
  - 通过管道执行的多个命令属于同一个进程组
  - 每个进程可通过setpgid()加入已有的进程组或创建一个新的进程组
- 会话是一个或者多个进程组的集合
  - 开启一个新的会话(setsid系统调用)的进程就是首进程(session leader)，而会话的ID就是开启新会话的首进程的ID
  - 通常用户登录时会开启一个会话，退出时结束会话。新会话的首进程就是bash进程
  - 一个会话可以有一个控制终端，会话首进程所打开的终端设备称为控制终端（可通过/dev/tty访问)。相应地会话首进程也称为控制进程(一般为shell进程)
  - 控制终端用来接收用户的输入以及输出到控制终端，发送终端组合键所触发的信号
  - 一个拥有控制终端的会话，有一个进程组为前台进程组(作业)，其他为后台进程组(作业)
  - **前台作业**由控制终端控制，**可以收终端的输入**，也可输出到控制终端，接收终端组合键触发的信号
  - **后台作业**独立于控制终端，它**无法接收终端的输入**，但是一般可以输出到控制终端
  - 执行命令时**尾部附加 &**，表示在后台执行该命令

进程组、会话和控制终端

- 用户登录后，登录shell为控制进程，位于前台，等待用户输入
- tty命令可查看连接的终端，可通过/dev/tty访问控制终端
- 执行外部命令时附加&，该作业后台运行
- 执行外部命令缺省前台运行，此时登录shell后台运行
- 前台作业允许读写控制终端
- 后台作业缺省允许写控制终端，但不允许读
- 控制终端产生的信号发送给前台作业
- 终端断开时发送信号SIGHUP给控制进程，控制进程再发送给各个作业，各个作业收到HUP信号时缺省终止进程



作业(job)控制

- bash支持作业控制，允许通过控制终端进行进程组的前台后台切换
- shell维护一个作业表纪录正在运行的作业(进程组)
- 一个作业被赋予一个作业号(job ID)，该作业号在通过shell控制的会话中唯一，从1开始递增
- 命令后加&，表示后台执行相应的命令，会输出一个作业ID和进程ID
  - 如果是一个管道命令，bash会输出管道中最后一个命令的进程ID
  - 如果是一个子shell，输出的是子shell的进程ID
- 后台作业缺省可写控制终端。stty tostop禁止后台作业写到控制终端，在写时会收到信号SIGTTOU，作业将被暂停
- 后台作业的结束停止等信息缺省会等待输出下一个提示符前才显示

```
dlmao@mars:~$ (sleep 5; echo sleep done)&
[1] 2118
dlmao@mars:~$ ps j
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 1485  1486  1486  1486 pts/0     2120 Ss    1000   0:00 -bash
 1486  2118  2118  1486 pts/0     2120 S     1000   0:00 -bash
 2118  2119  2118  1486 pts/0     2120 S     1000   0:00 sleep 5
 1486  2120  2120  1486 pts/0     2120 R+    1000   0:00 ps j
dlmao@mars:~$ sleep done

[1]+  Done                    ( sleep 5; echo sleep done )
dlmao@mars:~$
```

- ps j 查看进程状态，j选项表示显示作业管理相关字段

```
dlmao@mars:~$ stty tostop
dlmao@mars:~$ (sleep 5; echo done)&
[2] 2605
dlmao@mars:~$

[2]+  Stopped  ( sleep 5; echo done )
dlmao@mars:~$ # stty -tostop
```

作业控制命令

- 作业状态:
  - 前台运行: 正在运行的作业，可以从控制终端读写
  - 后台运行[Running]
  - 暂停运行[Stopped]：后台暂时停止，等待SIGCONT信号恢复执行
- 显示后台作业列表： jobs [options]　选项-l 显示进程ID
  - + 表示当前作业，即最近暂停的作业，如果没有，则最近切换到后台的那个作业
  - - 表示前一个作业
- 终端组合键Ctrl-Z发送SIGTSTP信号给前台作业，暂停该作业
- fg [jobspec] 将当前作业或者指定的作业切换到前台，发送SIGCONT信号恢复运行
- bg [jobspec] 将当前作业或者指定的作业切换到后台，发送SIGCONT信号恢复运行
- jobspec: 如何描述作业呢?
  - %job　作业号为job的作业，%也可省略(如bg 2)
  - %make　命令名前面为make的作业
  - %?game 命令中包含game的作业

```
dlmao@mars:~$ (sleep 300; echo done 300)&
[1] 2722
dlmao@mars:~$ (sleep 200; echo done 200)&
[2] 2724
dlmao@mars:~$ cat&
[3] 2726
dlmao@mars:~$ vi&
[4] 2727

[3]+  Stopped                 cat
dlmao@mars:~$ jobs -l
[1]   2722 Running ( sleep 300; echo done 300 ) &
[2]   2724 Running ( sleep 200; echo done 200 ) &
[3]-  2726 Stopped (tty input)     cat
[4]+  2727 Stopped (tty output)    vi
dlmao@mars:~$ fg
vi

[4]+  Stopped                 vi
dlmao@mars:~$ bg %4
[4]+ vi &
[4]+  Stopped                 vi
dlmao@mars:~$ echo $$ $!
1486 2727
dlmao@mars:~$ jobs -l
...
```

$$ 当前shell的pid
$! 最近切换到后台的进程pid

作业控制：nohup

- 控制终端断开(如网络连接断开）或用户输入exit或logout退出shell时，会话的控制终端要退出
  - 发送SIGHUP给控制进程，控制进程发送SIGHUP给前台作业
  - 如果有后台作业，会发送SIGHUP信号给所有的后台作业（一般收到SIGHUP信号会终止）
  - 守护进程(没有控制终端)在收到SIGHUP信号时一般重新读取配置文件
- shopt –s checkjobs打开checkjobs选项，输入exit或logout时不是马上退出，而是有后台作业时会提示有后台作业存在。继续退出时才发送SIGHUP信号给后台作业

nohup COMMAND [ARG]

- nohup命令会执行COMMAND，输入输出与终端脱离，**忽略SIGHUP信号**
- 如果其标准输入为终端，则重定向到/dev/null
- 如果错误输出为终端，重定向到标准输出
- 如果标准输出为终端，重定向到nohup.out
- 如果将nohup命令放到后台，在终端退出时也不会终止(忽略SIGHUP信号)

nohup cmd args &

```
dlmao@mars:~$ (sleep 3000; echo done)&
[3] 3788
dlmao@mars:~$ shopt -s checkjobs
dlmao@mars:~$ exit
logout
There are stopped jobs.
[3]   Running         ( sleep 3000; echo done ) &
dlmao@mars:~$ exit
logout
```

```
dlmao@mars:~$ nohup sleep 60 >log.txt &
[1] 3899
dlmao@mars:~$ nohup: ignoring input and redirecting
stderr to stdout
```

作业控制：disown

- 如果一个命令已经开始执行了，可通过jobs找到对应的作业号，然后使用bash内置命令disown –h %job命令
  - 表示在控制进程退出时不给其发送SIGHUP信号
  - 注意该作业的标准输入和输出并没有改变

disown [-h] [-ar] [jobspec ... | pid ...]

将指定的作业从作业列表中移走，注意进程本身并不会被kill

-h 不是从作业列表中移走，而是标记不给该作业发送SIGHUP信号

-a 如果后面参数没有时表示所有作业

-r 如果后面参数没有时表示所有状态为运行的作业

建议大家使用终端模拟器软件tmux(更加流行)或screen，这样只要Linux系统不关机，再次登录时仍然可回到以前的环境

查看进程状态ps(process status)

- 选项：BSD短选项(前无连字符) 、Unix短选项 (前有连字符) 和GNU长选项(前有两个连字符)

Unix短选项： ps [-aefFly] [-t tty] [-p pid] [-u userid]

BSD短选项： ps [ajluvx] [t tty] [p pid] [U userid]

查看哪些进程？缺省为当前用户且由当前控制终端控制的进程

- 当前用户还是所有用户的进程？
  - a 所有用户，终端为任一终端
- 由某个或者任意终端控制还是不要求由控制终端控制？
  - x 不要求终端

Unix选项	BSD选项	含义 （两种选项确定范围有时稍有区别）
ps	ps	**当前用户且当前控制终端**控制的进程，缺省
ps –a	ps a	所有用户(**a取消当前用户限制**)且和某个终端相关的进程。-a选项不包括session leader进程
ps –e/-A	ps ax	所有进程(包括守护进程)。**x取消终端限制**，包括守护进程
ps –p pidlist	ps p pidlist	与pidlist指定的进程ID相关的进程，以逗号分割
ps –u uidlist	ps U uidlist	与uidlist指定的用户相关的进程，以逗号分割
ps –t ttylist	ps t ttylist	与终端ttylist相关的进程， **-t - 表示没有控制终端**
ps -s sesslist		指定会话中的进程
ps –w	ps w	宽输出(不会截取字符)
ps –o format	ps o format	用户自定义输出的格式，比如 o uid,pid,user,args
ps –f/-F/-l/-ly		full-format/extra full format/long format/不显示flags
ps j/l/u		job control/long format(多了CPU详细信息)/**面向用户格式**
ps e		显示用到的环境变量 e=environment
ps jf		查看进程树**, j =job, f=forest**
ps –C cmdlist		指定命令名的相关进程，多个命令名之间以逗号分隔

查看进程状态ps

- **查看当前用户和当前终端进程 ps**
- **查看当前用户的所有进程ps x**
- 查看所有用户与终端相关的进程:
  - ps a、ps au 或者ps –af
- **查看所有进程：ps ax**或者 ps –e
  - 长格式： ps axj、ps aux、ps auxw
    或者ps –ef、ps –eF等
- **查看守护进程**（没有终端) ps –t -
- 查看进程树: ps axjf
- 查看某个服务相关的进程 ps -C sshd u

状态	含义
R	正在运行或等待运行
S	睡眠状态，可被唤醒；等待事件结束
T	停止状态，作业控制信号而挂起或traced
Z	僵尸进程，进程已结束但父进程不回收。命令之后有<defunct>
D	不可唤醒的睡眠状态，等待事件结束（磁盘I/O)

```
$ ps auxw
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.1 225452  9132 ?        Ss   May02   0:02 /sbin/init splash
root          2  0.0  0.0      0     0 ?        S    May02   0:00 [kthreadd]
root          4  0.0  0.0      0     0 ?        I<   May02   0:00 [kworker/0:0H]
root          6  0.0  0.0      0     0 ?        I<   May02   0:00 [mm_percpu_wq]
dlmao      4178  0.0  0.0  10876   840 pts/0    T    10:46   0:00 cat
dlmao      4182  0.0  0.0  40608  3360 pts/4    R+   10:47   0:00 ps axuw
```