进程概念

程序

存放在磁盘上的指令和数据的有序集合（文件）

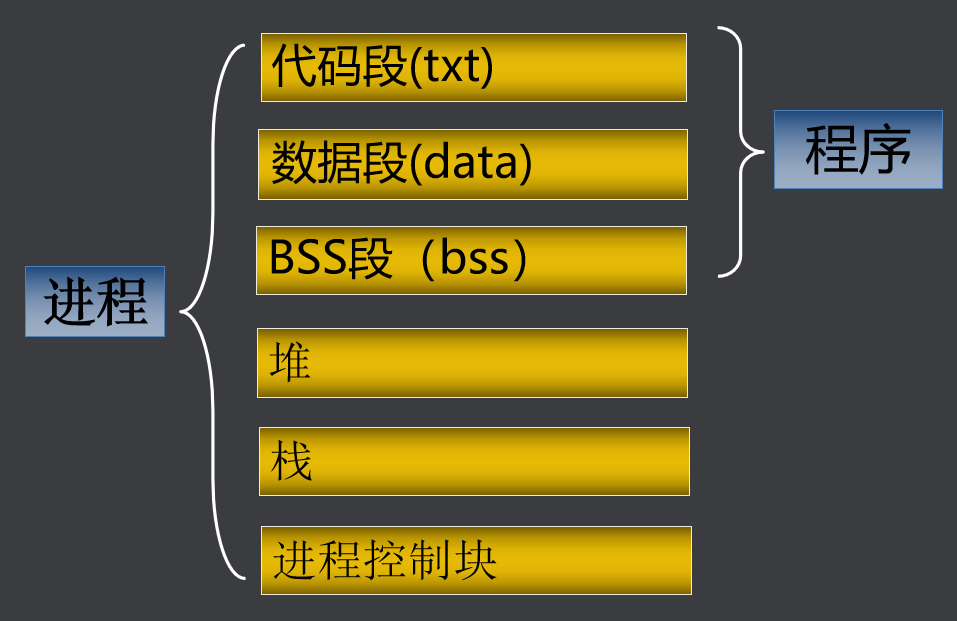
静态的

进程

执行一个程序所分配的资源的总称

动态的

进程和程序内容区别



进程包含的内容：

BSS段：存放程序中未初始化的全局变量

数据段：已初始化的全局变量

代码段：程序执行代码

堆（heap）：malloc等函数分配内存

栈(stack)：局部变量，函数参数，函数的返回值

进程控制块（pcb)：PID， 进程优先级，文件描述符表

进程类型

交互进程、批处理进程、守护进程。

进程状态

运行态、等待态、停止态、死亡态：

**查看进程信息**

ps 查看系统进程快照

top 查看进程动态信息

/proc 查看进程详细信息

ps 命令详细参数：

-e：显示所有进程

-l：长格式显示更加详细的信息

-f 全部列出，通常和其他选项联用

|  |  |
| --- | --- |
| **表头** | **含义** |
| F | 进程标志，说明进程的权限，常见的标志有两个:   * 1：进程可以被复制，但是不能被执行； * 4：进程使用超级用户权限； |
| S | 进程状态。进程状态。常见的状态有以下几种：   1. -D：不可被唤醒的睡眠状态，通常用于 I/O 情况。 2. -R：该进程正在运行。 3. -S：该进程处于睡眠状态，可被唤醒。 4. -T：停止状态，可能是在后台暂停或进程处于除错状态。 5. -W：内存交互状态（从 2.6 内核开始无效）。 6. -X：死掉的进程（应该不会出现）。 7. -Z：僵尸进程。进程已经中止，但是部分程序还在内存当中。 8. -<：高优先级（以下状态在 BSD 格式中出现）。 9. -N：低优先级。 10. -L：被锁入内存。 11. -s：包含子进程。 12. -l：多线程（小写 L）。 13. -+：位于后台。 |
| UID | 运行此进程的用户的 ID； |
| PID | 进程的 ID； |
| PPID | 父进程的 ID； |
| C | 该进程的 CPU 使用率，单位是百分比； |
| PRI | 进程的优先级，数值越小，该进程的优先级越高，越早被 CPU 执行； |
| NI | 进程的优先级，数值越小，该进程越早被执行； |
| ADDR | 该进程在内存的哪个位置； |
| SZ | 该进程占用多大内存； |
| WCHAN | 该进程是否运行。"-"代表正在运行； |
| TTY | 该进程由哪个终端产生； |
| TIME | 该进程占用 CPU 的运算时间，注意不是系统时间； |
| CMD | 产生此进程的命令名； |

top 查看进程动态信息

shift +> 后翻页

shift +< 前翻页

top -p PID 查看某个进程

**改变进程优先级**

nice 按用户指定的优先级运行进程

nice [-n NI值] 命令

NI 范围是 -20~19。数值越大优先级越低

普通用户调整 NI 值的范围是 0~19，而且只能调整自己的进程。

普通用户只能调高 NI 值，而不能降低。如原本 NI 值为 0，则只能调整为大于 0。

只有 root 用户才能设定进程 NI 值为负值，而且可以调整任何用户的进程。

renice 改变正在运行进程的优先级

renice [优先级] PID

jobs 查看后台进程

bg 将挂起的进程在后台运行

fg 把后台运行的进程放到前台运行

ctrl+z 把运行的前台进程转为后台并停止。

./test & 把test程序后台运行

创建子进程：

子进程概念：

#include <unistd.h>

pid\_t fork(void);

创建新的进程，失败时返回-1

成功时父进程返回子进程的进程号，子进程返回0

通过fork的返回值区分父进程和子进程

要点：1 子进程只执行fork之后的代码

1. 父子进程执行顺序是操作系统决定的。

子进程继承了父进程的内容

父子进程有独立的地址空间，互不影响

若父进程先结束

子进程成为孤儿进程，被init进程收养

子进程变成后台进程

若子进程先结束

父进程如果没有及时回收，子进程变成僵尸进程

进程的退出

#include <stdlib.h>

#include <unistd.h>

void exit(int status);

void \_exit(int status);

结束当前的进程并将status返回

exit结束进程时会刷新(流)缓冲区

return 和exit的区别

main函数结束时会隐式地调用exit函数，普通函数return是返回上一级。

进程的回收

#include <unistd.h>

pid\_t wait(int \*status);

成功时返回回收的子进程的进程号；失败时返回EOF

若子进程没有结束，父进程一直阻塞

若有多个子进程，哪个先结束就先回收

status 指定保存子进程返回值和结束方式的地址

status为NULL表示直接释放子进程PCB,不接收返回值

#include <unistd.h>

pid\_t waitpid(pid\_t pid, int \*status, int option);

参数：

pid

pid>0时，只等待进程ID等于pid的子进程，不管其它已经有多少子进程运行结束退出了，只要指定的子进程还没有结束，waitpid就会一直等下去。

pid=-1时，等待任何一个子进程退出，没有任何限制，此时waitpid和wait的作用一模一样。

pid=0时，等待同一个进程组中的任何子进程，如果子进程已经加入了别的进程组，waitpid不会对它做任何理睬。

pid<-1时，等待一个指定进程组中的任何子进程，这个进程组的ID等于pid的绝对值。

options

options提供了一些额外的选项来控制waitpid，目前在Linux中只支持WNOHANG和WUNTRACED两个选项，这是两个常数，可以用"|"运算符把它们连接起来使用

WNOHANG ：若由pid指定的子进程未发生状态改变(没有结束)，则waitpid()不阻塞，立即返回0

WUNTRACED： 返回终止子进程信息和因信号停止的子进程信息

wait(wait\_stat) 等价于waitpid(-1,wait\_stat,0)