

# Unix/Linux体系及编程

## Chap7 进程管理

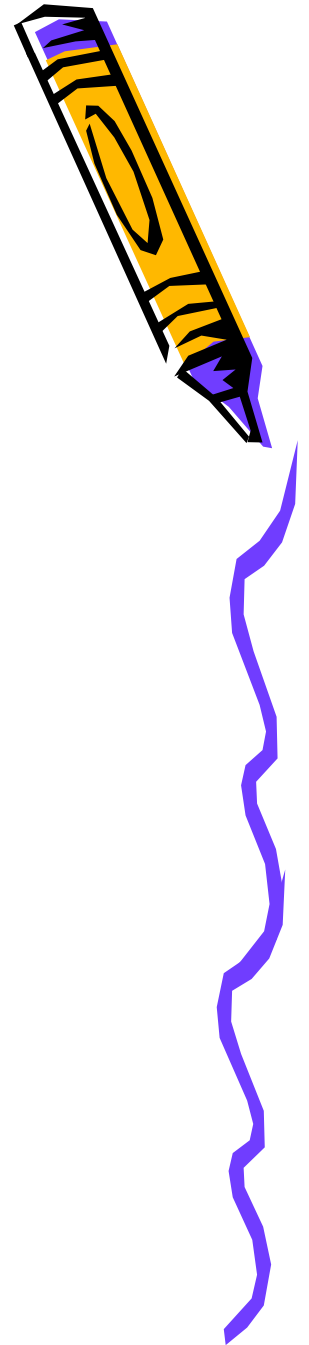
刘朝斌

zhbliu@gmail.com



# 主要内容

- 进程和多进程的概念
- 进程类型
- 如何运行后台进程
- 如何进行进程控制



# 7.1 进程概述

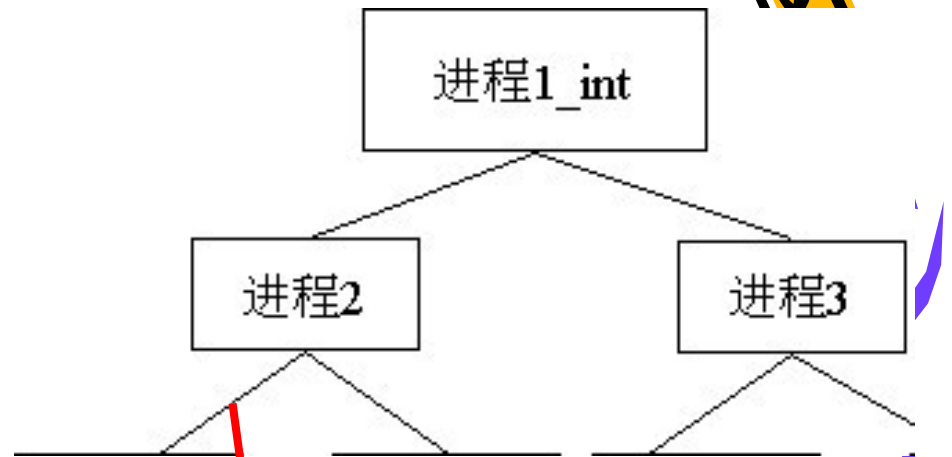
- 进程 (**process**)：正运行的程序，即程序只有在被系统载入了内存并运行后才能够叫做进程。
- 程序是磁盘文件，而进程则是内存中工作着的代码。
- 获得当前运行的程序及其进程号：**ps**

`[user@hostname]$ps [参数]`



## 7.1.2 进程间关系

- 复制机制来产生进程
- 进程关系：树形关系
- 每一个进程都记录了它的父进程和子进程的ID
- 进程结束之后退回到它的父进程
- 任何子进程都自动从父进程那里继承三个打开的设备文件：
  - 标准输入设备（键盘、鼠标）：**stdin**（**standard incoming**, 标准输入）
  - 标准输出设备（显示器等）：**stdout**（**standard outputting**, 标准输出）
  - ~~做~~标准错误输出（**stderr**, **standard error**）



**man init**  
**pstree**  
**pstree -up** show uid  
**pstree |grep bash**

grep 查看文件中符合的

## 7.2 进程类型

- 前台和后台进程

- 前台：一个程序控制着标准输入输出
- 后台：一个程序不从标准输入接受输入，一般也不将结果输出到标准输出

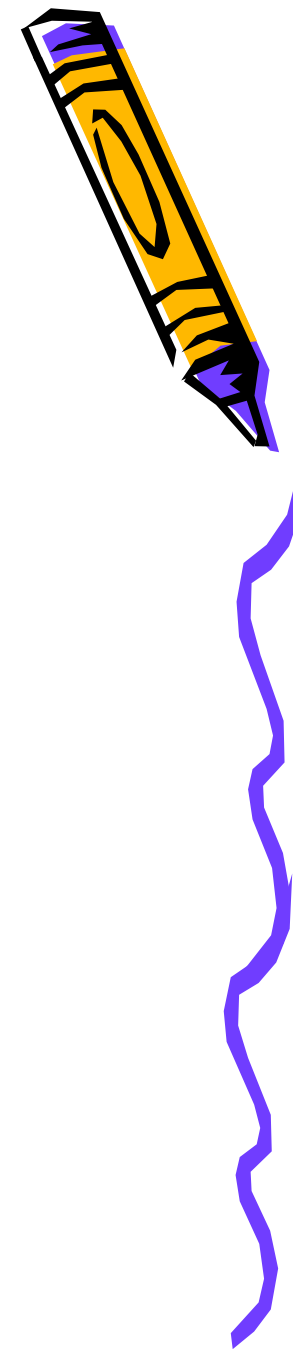
- 守护进程

- 一般以后台进程的方式存在
- 开机即被载入到系统中并常驻在系统中，直到关机时才结束，也被专称为“**Daemon**”，



## 7.3 进程控制命令

- 监视进程运行状态
- 在用户退出后让进程继续运行
- 更改进程的优先级
- 在进程有问题的时候杀死进程



# 监视进程

- 报告系统当前的进程状态:

**ps**

**# ps [选项]**

**ps -l**

显示所有包含其他使用者的行程

**ps aux**

以树形结构显示进程

**ps xjf**



USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.3	1336	432	?	S	08:09	0:04	init
root	2	0.0	0.0	0	0	?	SW	08:09	0:00	[keve
root	3	0.0	0.0	0	0	?	SW	08:09	0:00	[kapm
root	4	0.0	0.0	0	0	?	SWN	08:09	0:00	[ksof
root	5	0.0	0.0	0	0	?	SW	08:09	0:02	[kswa
root	6	0.0	0.0	0	0	?	SW	08:09	0:00	[bdf1
root	7	0.0	0.0	0	0	?	SW	08:09	0:00	[kupd
root	8	0.0	0.0	0	0	?	SW	08:09	0:00	[mdre
root	12	0.0	0.0	0	0	?	SW	08:09	0:00	[kjou
root	64	0.0	0.0	0	0	?	SW	08:09	0:00	[khub
root	156	0.0	0.0	0	0	?	SW	08:09	0:00	[kjou
root	410	0.0	0.0	0	0	?	SW	08:10	0:00	[eth0
root	466	0.0	0.4	1396	528	?	S	08:10	0:00	syslo
root	470	0.0	0.2	1336	360	?	S	08:10	0:00	klogd
rpc	481	0.0	0.3	1484	380	?	S	08:10	0:00	portm
rpcuser	500	0.0	0.3	1528	496	?	S	08:10	0:00	rpc.s

• **ps -aux --sort -pcpu | head**

Thursday, May 13, 2021

根据 CPU使用 来升序排序

默认为 10，即显示 10 行的内容。

7/24

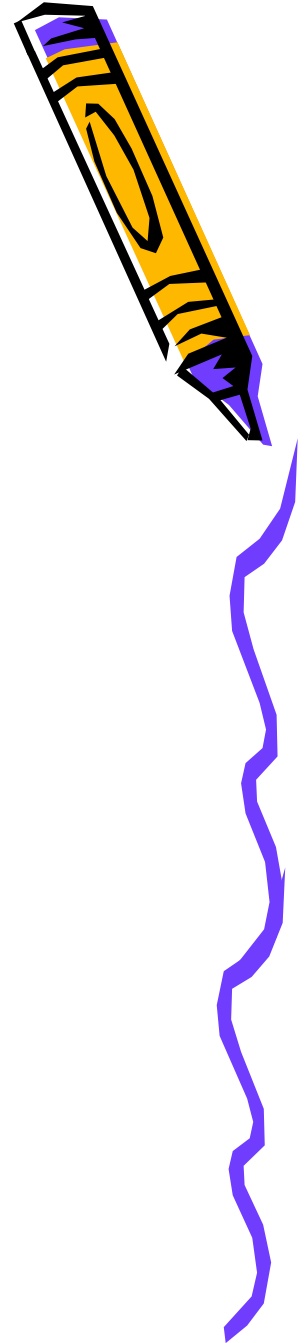
# top

Linux top命令用于实时显示 process 的动态。

使用权限：所有使用者。

- h //help
- u //user 查看哪个用户
- k //kill process
- cat /proc/loadavg
- man top //Linux memory types

```
top - display Linux processes
```



Thursday, May 13, 2021



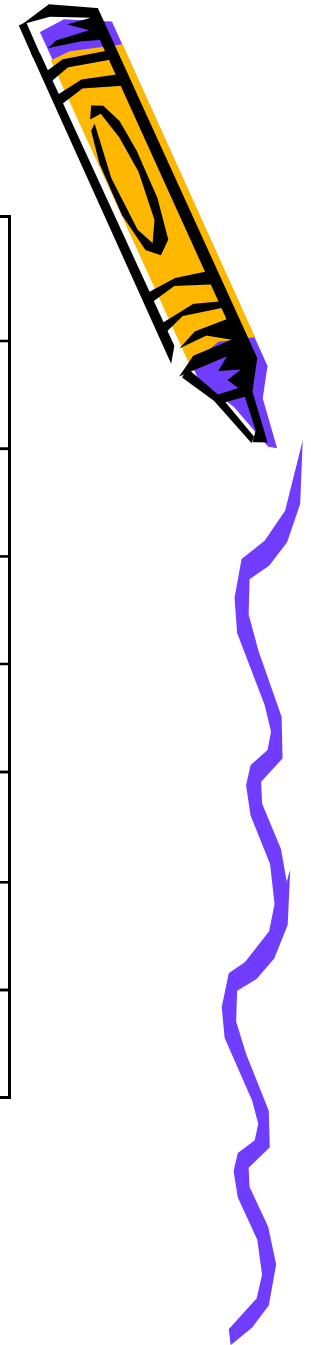
# ps -l 命令输出字段含义

字段名	意义
	1 不能run 0:能fork() 能run
<b>F</b>	process flags, 进程的权限。4 表示root ; 1 表示此进程仅能fork
<b>S</b>	进程的状态 (STAT)
<b>UID/PID/PPID</b>	该进程所属用户/进程号, 可以唯一标识该进程/父进程号
<b>C (%)</b>	进程自最近一次刷新以来所占用的CPU时间和总时间的百分比
<b>PRI/NI</b>	Priority/Nice; 进程的优先级值, 越小意味着越高的优先级
<b>ADDR</b>	该进程在内存的地址, 如果是个 running 进程, 一般就会显示 -
<b>SZ</b>	代表此进程用掉多少内存
<b>WCHAN</b>	进程等待的内核事件名, - 为运行
<b>TTY</b>	进程相关的终端 (远程登录为pts/n)
<b>TIME</b>	使用掉的 CPU 时间, 注意是实际花费掉的 CPU 运作的时间, 而不是系统时间;
<b>CMD</b>	被执行的命令行/此进程的指令

- STAT: 该行程的状态:
  - D: 无法中断的休眠状态 (通常 IO 的进程)
  - R: 正在执行中
  - S: 静止状态
  - T: 暂停执行

# 进程状态 **man ps**

符号	含义
R	运行或准备运行
S	睡眠状态
I	空闲 idlesse
Z	僵尸 zombie
D	不间断睡眠 disk sleep
W	进程没有驻留页
T	停止或跟踪 tracing stop



**fs/proc/array.c**

**include/linux/sched.h**

**// #131**

**// #108**

Thursday, May 13, 2021

# 调整优先级

- 在启动进程时指定优先级 (**man nice**)

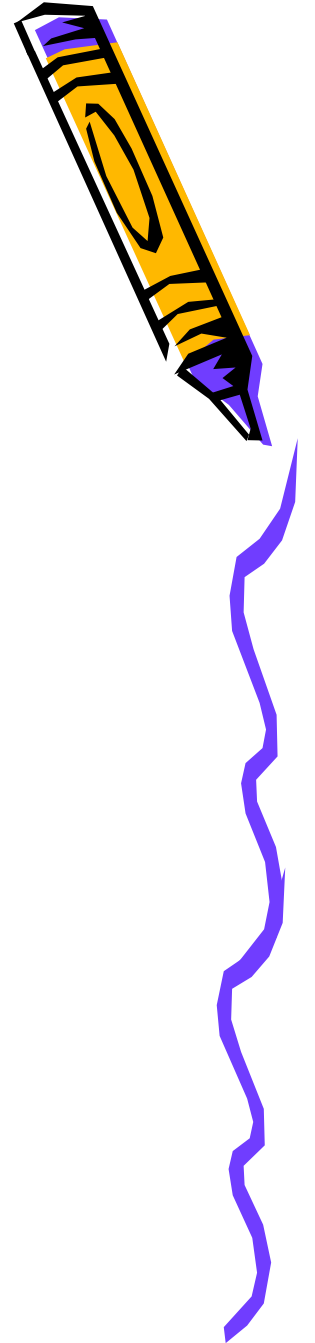
*nice* - 优先级改变量 命令[参数][对象]&

*默认显示当前的nice值*

- 进程运行时调整优先级

*renice* 优先级的改变量[PID][ -u用户...][ -p  
PID...][ -g GID...]

*nice -5 lp file1 &*



# 运行后台进程

- 使用**&**符号

意思是放到后台执行

```
[echo@echo echo]$ ls -R>dirlist &  
[1] 561
```

- 适合于：
  - 程序运行途中不需要用户的干预
  - 程序执行时间较长

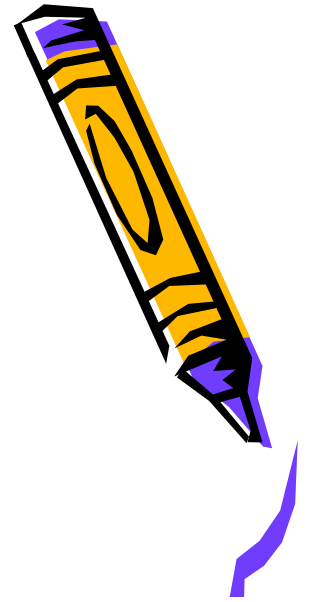
- 使用**nohup** 全称 no hang up (不挂起)，用于在系统后台不挂断地运行命令，退出终端不会影响程序的运行。  
**Nohup** 命令[参数]输出文件&
  - 程序的优先级增加5

- 使用**crontab** Linux crontab是用来定期执行程序的命令。

- 描述传递给**cron**的执行程序 crontab [-u user] file
- 格式：

*minute hour day month dayofweek command*

保存在 /var/spool/cron 目录



# 终止进程

- Ctrl+c: 终止前台程序 (返回值\$?)
- kill : 送一个结束进程的的信号到某个当前运行的特定进程, 从而结束进程

# kill [选项] [信号] 进程号

- 正常结束            kill PID
- 强制结束            kill -9 PID
- kill -l            若不加<信息编号>选项, 则 -l 参数会列出全部的信息名称。

- \$ watch -n 5 date&

监测一个命令 ( 这里为date )  
的运行结果

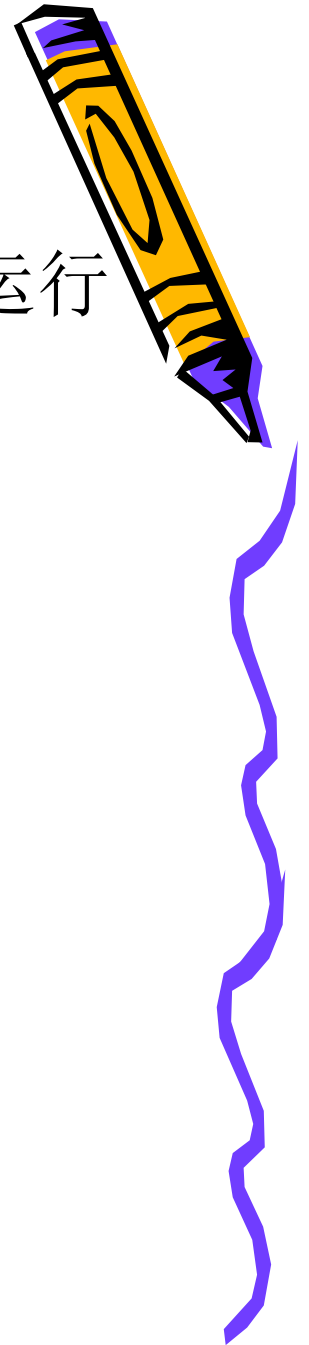
- \$ ps -l

- \$ kill PID

- \$ kill -9 PID

- \$ ps -l

- \$ lynx www.dlmu.edu.cn &

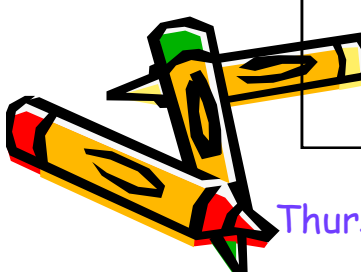



# jobs, bg, fg

- `ping www.dlmu.edu.cn`
  - `ctrl+z` //暂停当前进程放到后台，并且处于暂停状态
  - `jobs -l` //显示当前后台作业
  - `vi bg.txt`
  - `ctrl+z` //将vi进程转为后台作业
  - `jobs -l` 查看已挂载的程序,不用携带任何参数
  - `watch -n 10 date&`
  - `fg %jobnumber` 把进程挂到前台jobnumber的值时jobs -l查出来的
  - `kill -9 %jobnumber`
- fg %1



- **nohup**: 让用户的程序在用户退出系统后继续运行



```
[echo@echo echo]$nohup ls &
[1] 613
[echo@echo echo]$nohup: appending output to 'nohup.out'

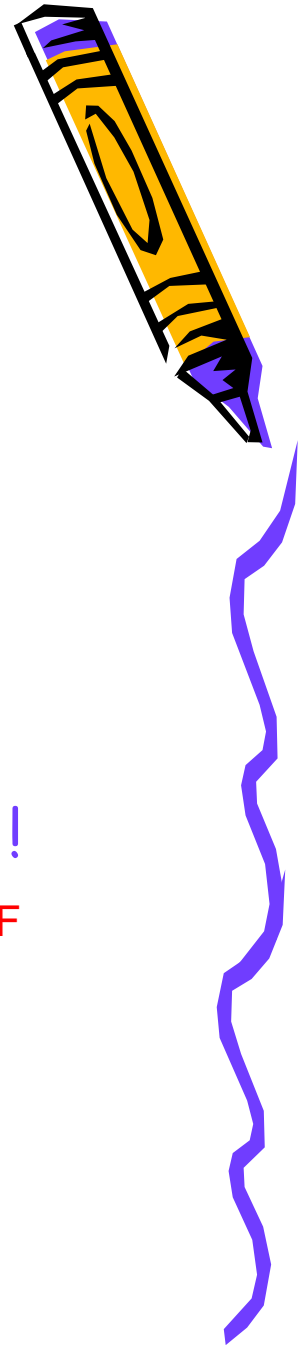
[1]+  Done                  nohup.dir
[echo@echo echo]$

[echo@echo echo]$ ls
#*mail*#1565tBp#  Desktop  Graphic  KDE1.gif  RMAIL  bin
dustbin          lsecho   nohup.out  notmal   work

[echo@echo echo]$ cat nohup.out
#*mail*#1565tBp#  Graphic  RMAIL  dustbin  normal
Desktop  KDE1.git  bin    nohup.out  work
$ nohup wget *** &
```

# queue, examine or delete jobs for later execution

- executes commands **at** a specified time.
  - \$ at 12:01 1 Jan 2012
  - \$ at 5pm + 3 days
  - \$ at 3:42am
  - \$ at now + 3 weeks
  - \$ at teatime
  - \$ at midnight
  - \$ at 23:59 12/24/2012 echo the end of world !
- \$ ctrl+d ctrl-d 不是发送信号，而是表示一个特殊的二进制值，表示 EOF
- 使用 **atq**：列出用户未执行完的任务
- 使用 **atrm**：删除后台执行的任务
- 使用 **batch**：在系统负载允许的情况下执行命令





# Scheduling Regularly Occurring Jobs with **cron**



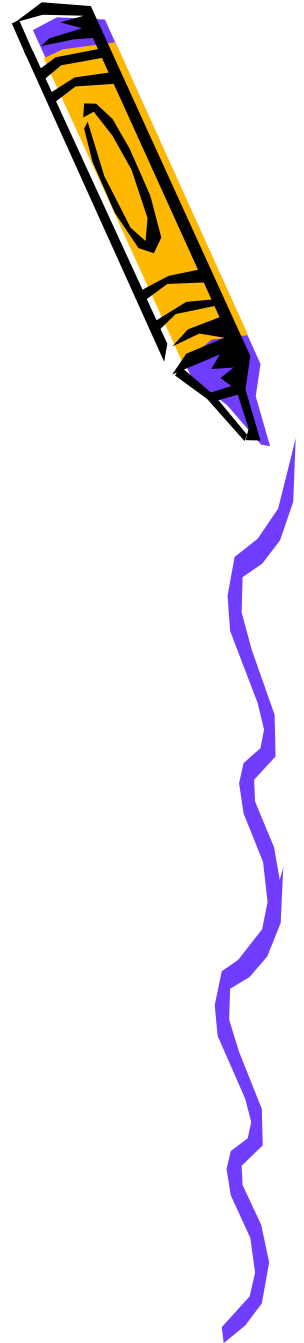
- `0 6 * * * echo "Good morning."`
  - minutes, hours, day, month, week
  - `1 * * * *`, it will happen at one minute after every hour
  - `15 3 * * *`, it will happen at 3:15 a.m. every day
  - `59 23 31 * *`, it will happen at 11:59 p.m., seven times a year (once in each of the months with a 31st)
  - `0 12 * * 0`, it will happen at noon on Sundays
- `$ crontab -l` -l: 列出目前的时程表
- `$ crontab -e` -e: 执行文字编辑器来设定时程表，内定的文字编辑器是 VI
- `55 16 * * 1-5 mail -s "Go home now!" buyhorse`



# 查询特殊进程示例

查询stopped进程

- `ps -e j | grep T`
- `ps -A -ostat,ppid,pid,cmd | grep -e '^[T]'`
- 查询zombie进程
- `ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]'`
- `kill -9 pid`



# /proc 目录

- 伪文件系统
- 通用信息: `cpuinfo`; `devices`; `diskstats`; `filesystems`; `partitions`; `interrupts`; `loadavg`; `locks`; `mdstat`; `meminfo`; `swaps`; `uptime`; `vmstat`; `version`; `modules`;
- 每个进程对应一个目录 `/proc/$pid`



# /proc/\$pid/fd

这个目录包含了进程打开的每一个文件的链接；



```
buyhorse@ubuntu-server-1804:/proc/13046$ ls -l fd
total 0
lrwx----- 1 buyhorse buyhorse 64 Apr 13 12:37 0 -> /dev/pts/2
lrwx----- 1 buyhorse buyhorse 64 Apr 13 12:37 1 -> /dev/pts/2
lrwx----- 1 buyhorse buyhorse 64 Apr 13 12:37 2 -> /dev/pts/2
lrwx----- 1 buyhorse buyhorse 64 Apr 13 12:48 255 -> /dev/pts/2
buyhorse@ubuntu-server-1804:/proc/13046$
```



Thursday, May 13, 2021

# /proc/\$pid/envIRON

- 二进制文件
- 包含该进程运行的环境变量
- `cat proc/$pid/envIRON`
- 可代替`getenv()`,`getpwd`等函数，自行读取该文件
- Eg: `getenv.c`




```
buyhorse@ubuntu:~$ ps
PID TTY      TIME CMD
 883 pts/0    00:00:00 bash
2244 pts/0    00:00:00 ps
buyhorse@ubuntu:~$ cd /proc/883/
buyhorse@ubuntu:/proc/883$ cat environ
LANG=en_HK.UTF-8USER=buyhorseLOGNAME=buyhorseHOME=/home/buyhorsePATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
MAIL=/var/mail/buyhorseSHELL=/bin/bashSSH_CLIENT=172.27.69.196 52098 22SSH_CONNECTION=172.27.69.196 52098 172.27.70.148 22SSH_TTY=/dev/pts/0TERM=xtermXDG_SES
SION_ID=45815XDG_RUNTIME_DIR=/run/user/1000JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64LANGUAGE=en_HK:enbuyhorse@ubuntu:/proc/883$
```

# /proc/\$pid/cwd /proc/\$pid/exe


指向该进程相应的可  
执行文件及其路径

cmdline.c



```
1 #include "stdio.h"
2 #include "wait.h"
3 int
4 main ()
5 {
6 {
7 int i;
8     for(i=0;i<100;i++) {
9         printf("Hello World!\n");
10        sleep(3);
11    }
12 }
13 return 0;
14 }
```

buyhorse@ubuntu-server-1804: /proc/12894




```
buyhorse@ubuntu-server-1804:/proc$ ps -A |grep a.out
12894 pts/1    00:00:00 a.out
buyhorse@ubuntu-server-1804:/proc$ cd 12894/
buyhorse@ubuntu-server-1804:/proc/12894$ ll cwd
lrwxrwxrwx 1 buyhorse buyhorse 0 Apr 13 11:21 cwd -> /home/buyhorse/unix/
buyhorse@ubuntu-server-1804:/proc/12894$ ll exe
lrwxrwxrwx 1 buyhorse buyhorse 0 Apr 13 11:21 exe -> /home/buyhorse/unix/a.out*
buyhorse@ubuntu-server-1804:/proc/12894$
```

Thursday, May 13, 2021



# /proc/\$pid/cmdline

- cmdline.c
- \$ ./a.out tom jerry



```
buyhorse@ubuntu:~/apue$ ps -A |grep a.out
21219 pts/1    00:00:00 a.out
buyhorse@ubuntu:~/apue$ cd /proc/21219
buyhorse@ubuntu:/proc/21219$ cat cmdline
./a.outtomjerrybuyhorse@ubuntu:/proc/21219$ cp cmdline ~/cmdline
buyhorse@ubuntu:/proc/21219$ cd
buyhorse@ubuntu:~$ cat cmdline
./a.outtomjerrybuyhorse@ubuntu:~$ vi cmdline
buyhorse@ubuntu:~$ cat cmdline
./a.outtomjerrybuyhorse@ubuntu:~$ od -c cmdline
0000000  .   /   a   .   o   u   t   \0   t   o   m   \0   j   e   r   r
0000020  y   \0
0000022
buyhorse@ubuntu:~$ █
```



```
1 █ ./a.out^@tom^@jerry^@
```

# 小结&习题

- 思考题

- (1) 什么是进程？
- (2) 如何显示进程？
- (3) 进程之间具有什么样的关系？
- (4) 什么是多进程和多任务？

- 上机题

- (1) 查看系统运行后台进程。
- (2) 如何启动多个进程？
- (3) 如何调整进程的优先级？
- (4) 进程相关命令

