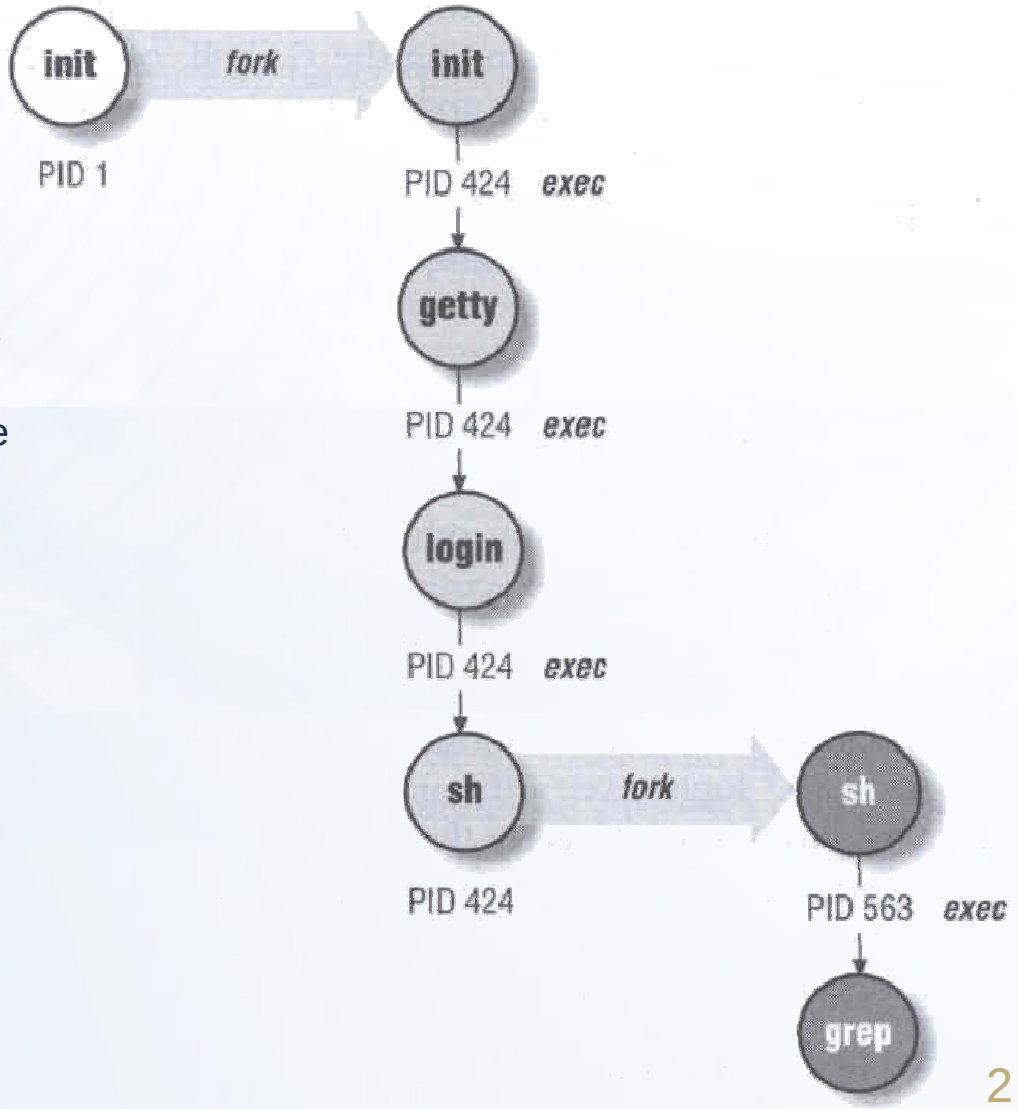
Chapter 4
Controlling Processes

Program to Process

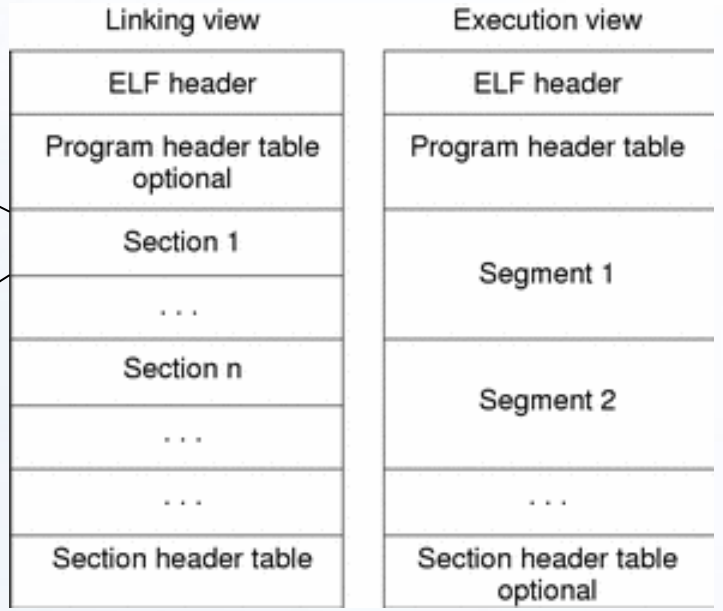
- Program is dead
  - Just lie on disk
  - grep is a program
    - **/usr/bin/grep**
    - **% file /usr/bin/grep**
      - ELF 32-bit LSB executable
- When you execute it
  - It becomes a process
- Process is alive
  - It resides in memory



ELF

> Executable and Linkable Format

.text
    CPU instructions
.bss
    un-initialized data
.data
    initialized data



Into Memory

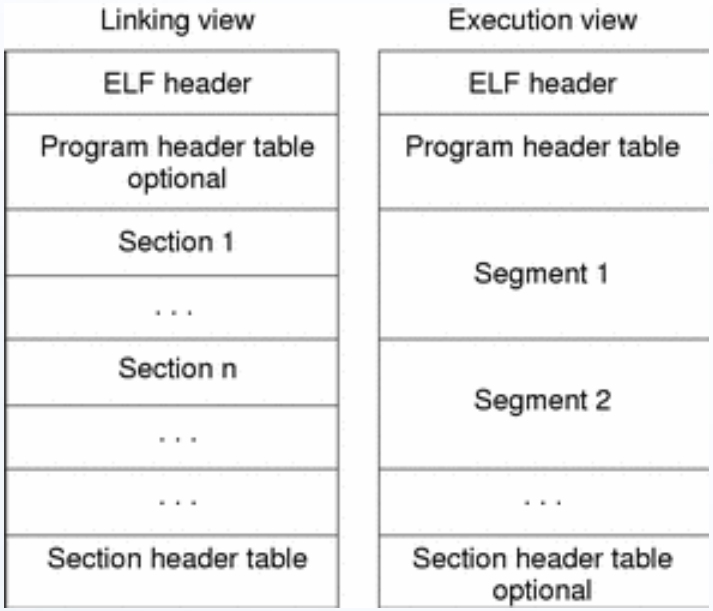
Components of a Process

- An address space in memory
  - Code and data of this process
- A set of data structures within the kernel
  - Used to monitor, schedule, trace, …., this process
    - **owner**
    - **Current status**
    - **Execution priority**
    - **Information of used resource**
    - **Signal mask**

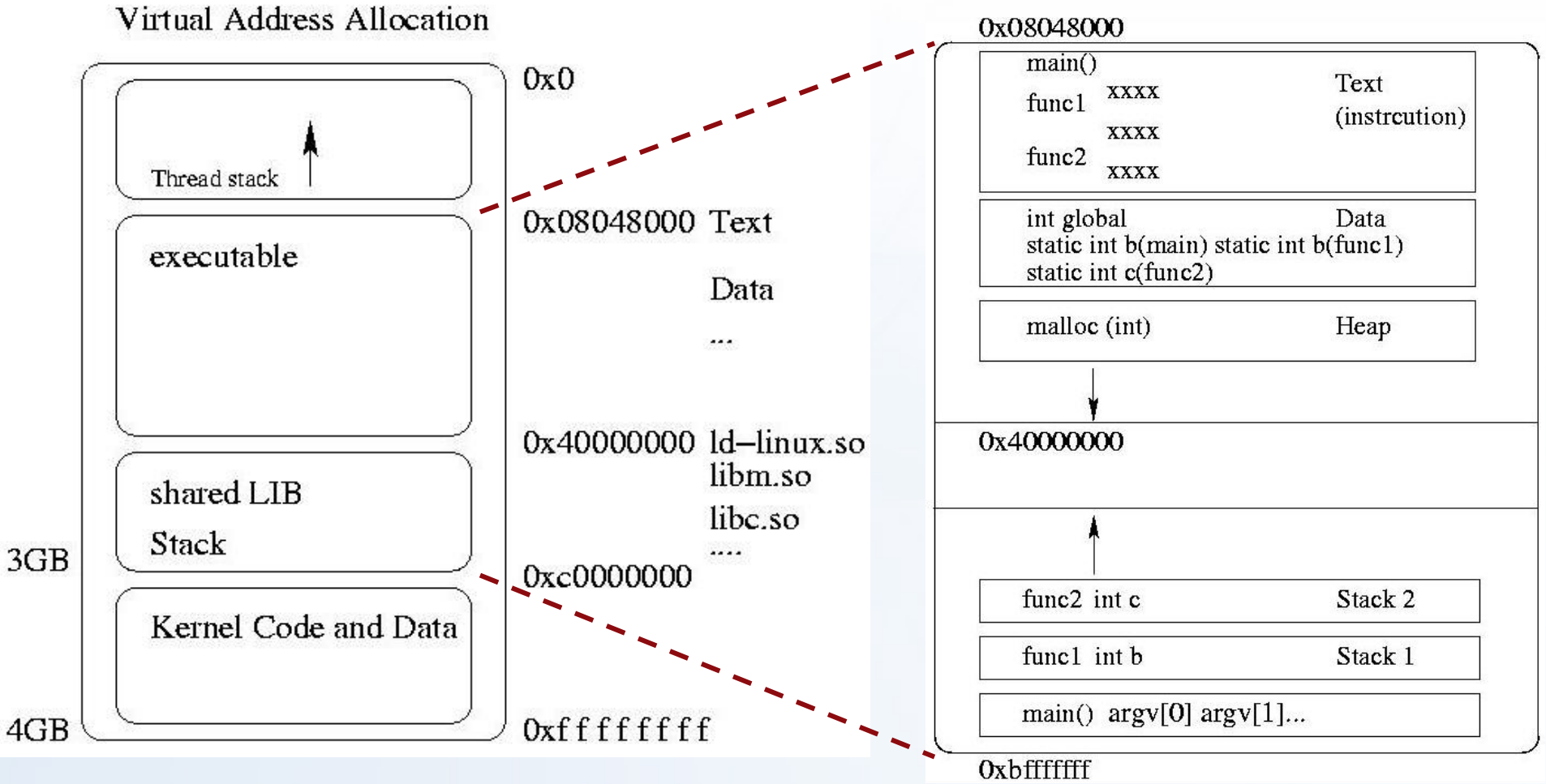
Components of a Process – address space in memory (1)

```
1  int global;
2  static int func1 ()
3  {
4                  static int b;
5                  int *c;
6                  int d;
7                  func2();
8                  return 1;
9  }
10 int func2 ()
11 {
12                 int c;
13                 static int d;
14                 return 2;
15 }
16 int main()
17 {
18                 int a;
19                 static int b;
20                 int init = 3;
21                 func1();
22                 return 3;
23 }
```

gcc

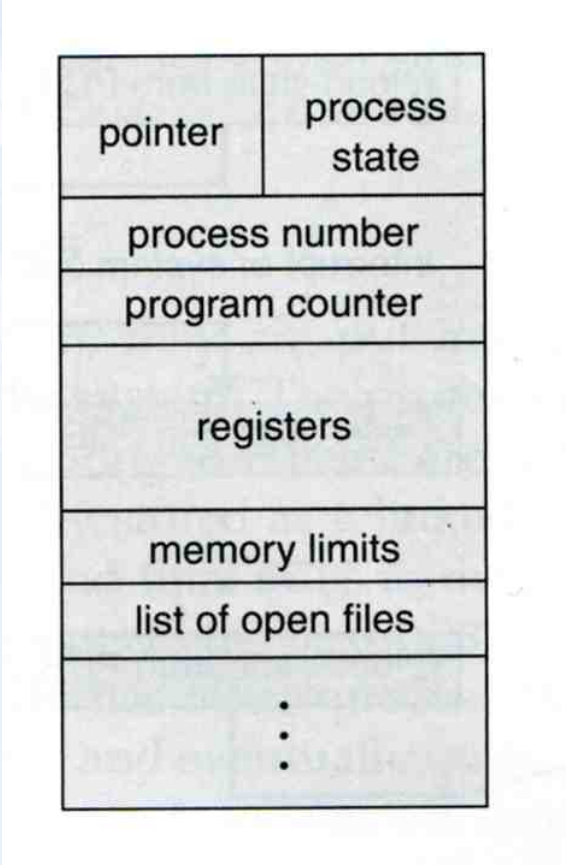


Components of a Process – address space in memory (2)



Components of a Process – data structure in kernel (1)

- In OS, we call these as PCB
  - Process Control Block



Components of a Process – data structure in kernel (2)

> FreeBSD

- /usr/include/sys/proc.h
- struct proc

```
struct proc
{
        …
        pid_t p_pid;
        struct pcred *p_cred;
        …
}
struct pcred
{
        …
        uid_t p_ruid, p_svuid;
        gid_t p_rgid,p_svgid;
        …
}
```

> Linux

- /usr/include/linux/sched.h
- struct task_struct

```
struct task_struct
{
        …
        pid_t pid;
        pid_t pgrp;
        uid_t uid,euid,suid,fsuid;
        gid_t gid,egid,sgid,fsgid;
        char comm[16];
        …
}
```

Attributes of the Process

- PID, PPID
  - Process ID and parent process ID
- UID, EUID
  - User ID and Effective user ID
- GID, EGID
  - Group ID and Effective group ID
- Niceness
  - The suggested priority of this process

Attributes of the process – PID and PPID

- PID – process id
  - Unique number assigned for each process in increasing order when they are created
- PPID – parent PID
  - The PID of the parent from which it was cloned
  - UNIX uses fork-and-exec model to create new process

```c
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    int pid,i;

    pid = fork();
    if (pid == 0) {
        for (i=0;i<12;i++) {
            printf("I am a child process, my pid is %d, parent pid is %d\n",getpid(),getppid());
            sleep(1);
        }
        exit(1);
    }
    else if (pid > 0) {
        for (i=0;i<10;i++) {
            printf(" I am a parent process, my pid is %d, parent pid is %d\n",getpid(),getppid());
            sleep(1);
        }
    }
    else if (pid < 0)
        printf(" Sorry .....I can't fork my self\n");

    return 0;
}
```

Attributes of the process – UID、GID、EUID and EGID

> UID, GID, EUID, EGID

- The effective uid and gid can be used to enable or restrict the additional permissions
- Effective uid will be set to
  - **Real uid if setuid bit is off**
  - **The file owner's uid if setuid bit is on**

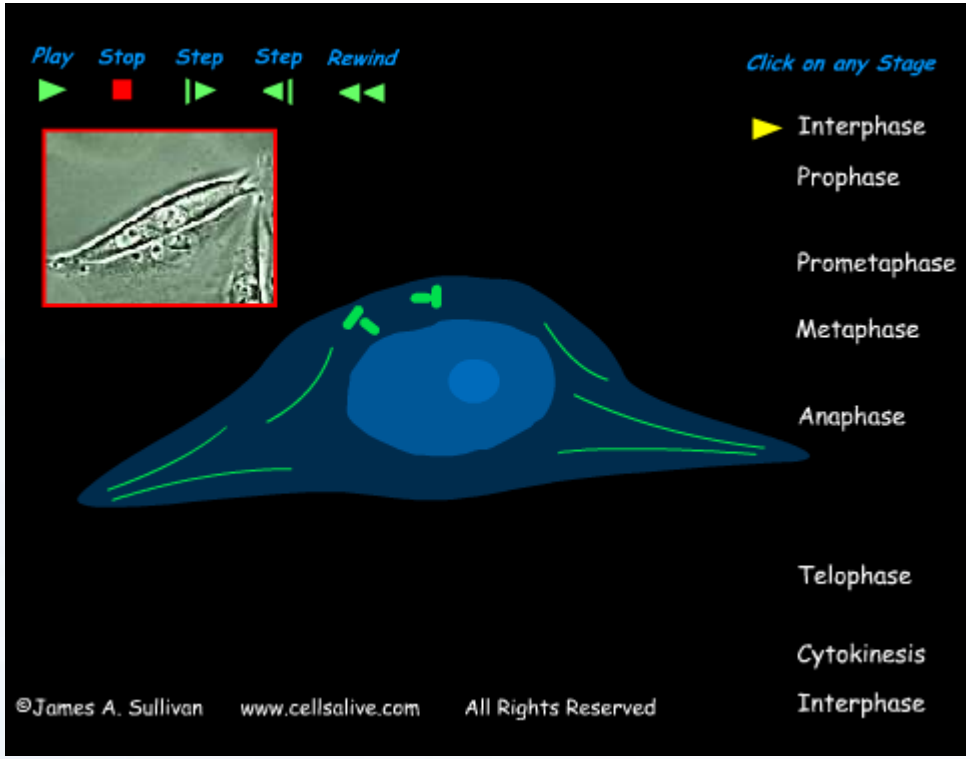
Ex:

/etc/master.passwd is "root read-write only" and /usr/bin/passwd is a "setuid" program

```
tytsai@tytsai [6:31pm] /etc> ls -al | grep passwd
-rw-------   1 root  wheel      1337 Sep 18 17:28 master.passwd
-rw-r--r--   1 root  wheel      1171 Sep 18 17:28 passwd
tytsai@tytsai [6:31pm] /etc> ls -al /usr/bin/ | grep passwd
-r-sr-xr-x   1 root  wheel     10520 Apr  3  2003 opiepasswd*
-r-sr-xr-x   2 root  wheel     28828 Apr  3  2003 passwd*
-r-sr-xr-x   2 root  wheel     28828 Apr  3  2003 yppasswd*
tytsai@tytsai [6:31pm] /etc>
```

Process Lifecycle

- fork
  - child has the same program context
- exec
  - child use exec to change the program context
- exit
  - child use _exit to tell kernel that it is ready to die and this death should be acknowledged by the child's parent
- wait
  - parent use wait to wait for child's death
  - If parent died before child, this orphan process will have init as it's new parent



Signal

- A way of telling a process something has happened
- Signals can be sent
  - among processes as a means of communication
  - by the terminal driver to kill, interrupt, or suspend process
    - **<Ctrl-C>、<Ctrl-Z>**
  - by the administrator to achieve various results
  - by the kernel when a process violate the rules, such as divide by zero

Actions when receiving signal

- Depend on whether there is a designated handler routine for that signal
  1. If yes, the handler is called
  2. If no, the kernel takes some default action
- "Catching" the signal
  - Specify a handler routine for a signal within a program
- Two ways to prevent signals from arriving
  1. Ignored
     - **Just discard it and there is no effect to process**
  2. Blocked
     - **Queue for delivery until unblocked**
     - **The handler for a newly unblocked signal is called only once**

UNIX signals

> man signal or see /usr/include/sys/signal.h

FreeBSD

#	Name	Description	Default	Catch	Block	Dump core
1	SIGHUP	Hangup	Terminate	☑	☑	⊘
2	SIGINT	Interrupt (^C)	Terminate	☑	☑	⊘
3	SIGQUIT	Quit	Terminate	☑	☑	☑
9	SIGKILL	Kill	Terminate	⊘	⊘	⊘
10	SIGBUS	Bus error	Terminate	☑	☑	☑
11	SIGSEGV	Segmentation fault	Terminate	☑	☑	☑
15	SIGTERM	Soft. termination	Terminate	☑	☑	⊘
17	SIGSTOP	Stop	Stop	⊘	⊘	⊘
18	SIGTSTP	Stop from tty (^Z)	Stop	☑	☑	⊘
19	SIGCONT	Continue after stop	Ignore	☑	⊘	⊘

Kill – send signals

- % kill [-signal] pid
  - Ex:
    - **First, find out the pid you want to kill**
    - **% kill –l (list all available signals)**
    - **% kill 49222**
    - **% kill –TERM 49222**
    - **% kill –15 49222**

Process States

- man ps and see "state" keyword

State	Meaning
I	Idle
R	Runnable
S	Sleeping
T	Stopped
Z	Zombie
D	in Disk

Niceness

- How kindly of you when contending CPU time
  - High nice value ➔ low priority
- Inherent Property
  - A newly created process inherits the nice value of its parent
    - **Prevent processes with low priority from bearing high-priority children**
- Root has complete freedom in setting nice value
  - Use nice to start a high-priority shell to beat berserk process

nice and renice commands

> nice format

- OS nice : % /usr/bin/nice [range] utility [argument]
- csh nice : % nice [range] utility [argument]
  - **% nice +10 ps -l**

> renice format

- % renice [prio | -n incr] [–p pid] [–g gid] [-u user]
  - **% renice 15 –u tytsai**

System	Prio. Range	OS nice	csh nice	renice
FreeBSD	-20 ~ 20	-incr | -n incr	+prio | -prio	prio | -n incr
Red Hat	-20 ~ 20	-incr | -n incr	+prio | -prio	prio
Solaris	0 ~ 39	-incr | -n incr	+incr | -incr	prio | -n incr
SunOS	-20 ~ 19	-incr	+prio | -prio	prio

ps command (BSD、Linux)

> ps

```
tytsai@tybsd:~> ps
  PID  TT  STAT      TIME COMMAND
  125  p0  Ss     0:00.03 -tcsh (tcsh)
  139  p0  R+     0:00.00 ps
```

> ps aux

```
tytsai@tybsd:~> ps aux
USER       PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED     TIME COMMAND
root         0  0.0  0.0     0    0  ??  DLs  4:50PM  0:00.00 (swapper)
tytsai     125  0.0  0.2  1456 1128  p0   Ss   8:52AM  0:00.04 -tcsh (tcsh)
tytsai     124  0.0  0.4  5296 2304  ??   S    8:52AM  0:00.01 sshd: tytsai@ttyp
root        89  0.0  0.4  3144 2324  ??   Ss   8:50AM  0:00.02 sendmail: accepti
```

> ps auxww

```
tytsai@tybsd:~> ps auxww
USER       PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED     TIME COMMAND
root        89  0.0  0.4  3144 2324  ??   Ss   8:50AM  0:00.02 sendmail: accepti
g connections (sendmail)
```

ps command – Explanation of ps –aux (BSD、Linux)

Field	Contents
USER	Username of the process's owner
PID	Process ID
%CPU	Percentage of the CPU this process is using
%MEM	Percentage of real memory this process is using
VSZ	Virtual size of the process, in kilobytes
RSS	Resident set size (number of 1K pages in memory)
TT	Control terminal ID
STAT	Current process status: R = Runnable D = In disk (or short-term) wait I = Sleeping (> 20 sec) S = Sleeping (< 20 sec) T = Stopped Z = Zombie Additional Flags: > = Process has higher than normal priority N = Process has lower than normal priority < = Process is exceeding soft limit on memory use A = Process has requested random page replacement S = Process has asked for FIFO page replacement V = Process is suspended during a **vfork** E = Process is trying to **exit** L = Some pages are locked in core X = Process is being traced or debugged s = Process is a session leader (head of control terminal) W = Process is swapped out + = Process is in the foreground of its control terminal
STARTED	Time the process was started
TIME	CPU time the process has consumed
COMMAND	Command name and arguments[a]

ps command (BSD、Linux)

> ps –j

```
tytsai@tybsd:~> ps -j
USER    PID  PPID  PGID     SESS  JOBC  STAT  TT      TIME COMMAND
tytsai  125   124   125 c2053500     0    Ss  p0   0:00.06 -tcsh (tcsh)
tytsai  205   125   205 c2053500     1    R+  p0   0:00.00 ps -j
```

> ps –o

```
tytsai@tybsd:~> ps -o uid,pid,ppid,%cpu,%mem,command
  UID   PID  PPID %CPU %MEM COMMAND
 1001   125   124  0.0  0.2 -tcsh (tcsh)
 1001   237   125  0.0  0.1 ps -o uid,pid,ppid,%cpu,%mem,command
```

> ps -L

```
tytsai@tybsd:~> ps -L
%cpu %mem acflag acflg blocked caught command cpu cputime f flags ignored inblk
inblock jobc ktrace ktracep lim login logname lstart majflt minflt msgrcv msgsnd
ni nice nivcsw nsignals nsigs nswap nvcsw nwchan oublk oublock p_ru paddr pagein
pcpu pending pgid pid pmem ppid pri re rgid rlink rss rssize rsz rtprio ruid
ruser sess sig sigcatch sigignore sigmask sl start stat state svgid svuid tdev
time tpgid tsess tsiz tt tty ucomm uid upr user usrpri vsize vsz wchan xstat
```

top command

```
last pid: 49993;  load averages:  0.00,  0.01,  0.00   up 20+06:54:04  21:25:22
59 processes:  1 running, 58 sleeping
CPU states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
Mem: 30M Active, 265M Inact, 153M Wired, 48K Cache, 199M Buf, 1562M Free
Swap: 1024M Total, 1024M Free

  PID USERNAME   PRI NICE  SIZE    RES STATE  C   TIME   WCPU    CPU COMMAND
  201 mysql       2   0 36640K 16660K poll   0   4:45  0.00%  0.00% mysqld
  129 root        2   0   596K   360K select 0   2:22  0.00%  0.00% natd
  205 root        2   0  3164K  2492K select 0   0:38  0.00%  0.00% httpd
```

- Various usage
  - top –q — run top and renice it to -20
  - top –u — don't map uid to username
  - top –U*username* — show process owned by user
- Interactive command
  - o — change display order
  - u — show only processes owned by user

Runaway process

- Processes that use up excessive system resource or just go berserk
  - kill –STOP for unknown process
  - renice it to a higher nice value for reasonable process