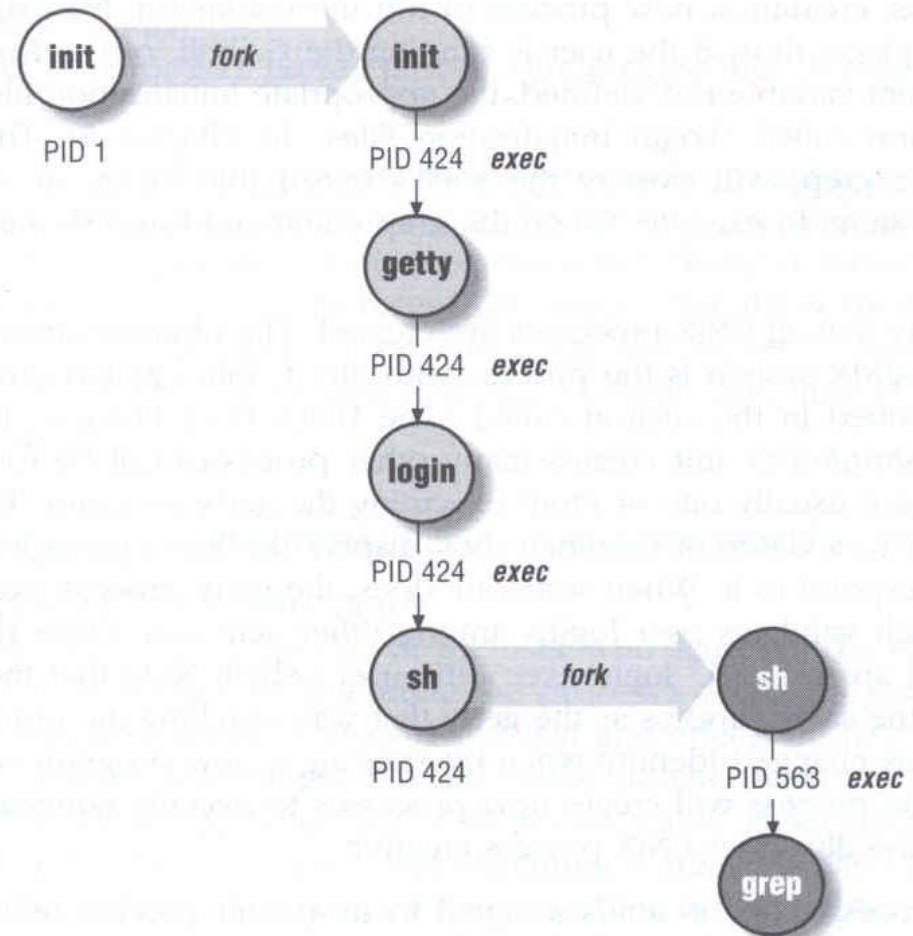


# Chapter 4

## 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



# 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, Group (Credentials)
    - Current status
    - VM space
    - Execution priority (scheduling info)
    - Information of used resource
    - Resource limits
    - Syscall vector
    - Signal actions

# Process Credentials

- ◎ 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

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main(void)
5 {
6     int pid,i;
7
8     pid = fork();
9     if (pid == 0) {
10         for (i=0;i<12;i++) {
11             printf("I am a child process, my pid is %d, parent pid is %d\n",getpid(),getppid());
12             sleep(1);
13         }
14         exit(1);
15     }
16     else if (pid > 0) {
17         for (i=0;i<10;i++) {
18             printf(" I am a parent process, my pid is %d, parent pid is %d\n",getpid(),getppid());
19             sleep(1);
20         }
21     }
22     else if (pid < 0)
23         printf(" Sorry .....I can't fork my self\n");
24
25     return 0;
26 }
```

# 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 root” program

```
lwbsd:~ -lwshsu- ls -al /etc/passwd /etc/master.passwd
-rw----- 1 root wheel - 1999 Sep 8 20:49 /etc/master.passwd
-rw-r--r-- 1 root wheel - 1727 Sep 8 20:49 /etc/passwd

lwbsd:~ -lwshsu- ls -al /usr/bin/passwd
-r-sr-xr-x 2 root wheel schg 8120 Sep 26 16:23 /usr/bin/passwd
```

# Process Lifecycle

- fork
  - > child has the same program context – fork(2)
- exec
  - > child use exec to change the program context – execve(2)
- exit
  - > child use \_exit to tell kernel that it is ready to die and this death should be acknowledged by the child's parent – \_exec(2)
- 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 – wait(2)

# 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

































# Signal –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

# Signal – BSD signals

- signal(3)
- /usr/include/sys/signal.h

#	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			

# Signal – Send signals: kill

- ◎ Kill(1) --terminate or signal a process
- ◎ \$ kill [-signal] pid
  - > Ex:
    - First, find out the pid you want to kill (ps, top, sockstat, lsof...)
    - % kill -l (list all available signals)
    - % kill 49222
    - % kill -TERM 49222
    - % kill -15 49222
  - > killall(1)
    - kill processes by name

# Process States

- ◉ man ps and see "state" keyword

State	Meaning
I	<b>I</b> dle
R	<b>R</b> unnable
S	<b>S</b> leeping
T	<b>S</b> topped
Z	<b>Z</b> ombie
D	in <b>D</b> isk wait

# 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

# Niceness – nice and renice

## ○ 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 lwhsu

System	Nice. 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

```
lwbsd:~ -lwshu- ps
  PID  TT  STAT      TIME COMMAND
68272  0  Ss+    0:00.05 -tcsh (tcsh)
54245  3  R+     0:00.00 ps
```

◎ ps aux

```
lucky7:~ -lwshu- ps aux
USER      PID %CPU %MEM    VSZ   RSS  TT  STAT  STARTED      TIME COMMAND
root         10 100.0  0.0      0     16  ??  RL    Sat03PM 4724:11.63 [idle: cpu1]
root         11  96.5  0.0      0     16  ??  RL    Sat03PM 4728:04.35 [idle: cpu0]
cvsup     63790   0.4  0.1   9056  4764  ??  S      2:59AM   0:14.86 /usr/local/sbin/cvsupd -e -C
16 -l @daemon -b /usr/local/etc/
lwshu     65013   0.4  0.1  11080  4176  p4  Ds     3:19AM   0:00.19 -tcsh (tcsh)
(...)
```

◎ ps auxww

```
lucky7:~ -lwshu- ps auxww | head
USER      PID %CPU %MEM    VSZ   RSS  TT  STAT  STARTED      TIME COMMAND
root         11  89.7  0.0      0     16  ??  RL    Sat03PM 4730:25.76 [idle: cpu0]
root         10  83.7  0.0      0     16  ??  RL    Sat03PM 4726:29.66 [idle: cpu1]
cvsup     65024  28.8  0.1   8212  4052  ??  R      3:20AM   0:17.07 /usr/local/sbin/cvsupd -e -C
16 -l @daemon -b /usr/local/etc/cvsupd -s sup.client
cvsup     63790   1.4  0.1   9056  4764  ??  D      2:59AM   0:16.87 /usr/local/sbin/cvsupd -e -C
16 -l @daemon -b /usr/local/etc/cvsupd -s sup.client
(...)
```

# 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 <b>vfork</b> E = Process is trying to <b>exit</b> 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 <sup>a</sup>



# ps command (BSD 、 Linux)

## ○ ps -j

Use these options with shell scripts

```
lucky7:~ -lwhsu- ps -j
USER      PID  PPID  PGID   SID  JOBC  STAT  TT      TIME  COMMAND
lwhsu    28905 28903 28905 28905    0  Is+   p0      0:00.16 -tcsh (tcsh)
lwhsu    65063 65013 65063 65013    1  R+    p4      0:00.00 ps -j
```

## ○ ps -o

```
lucky7:~ -lwhsu- ps -o uid,pid,ppid,%cpu,%mem,command
  UID   PID  PPID  %CPU  %MEM  COMMAND
  1000 28905 28903   0.0   0.0  -tcsh (tcsh)
  1000 30617 30615   0.0   0.0  -tcsh (tcsh)
  1000 65066 65013   0.0   0.0  ps -o uid,pid,ppid,%cpu,%mem,command
```

## ○ ps -L

```
lucky7:~ -lwhsu- ps -L
%cpu %mem acflag acflg args blocked caught comm command cpu cputime emul
etime f flags ignored inblk inblock jid jobc ktrace label lim lockname
login logname lstart lwp majflt minflt msgrcv msgsnd mwchan ni nice
nivcsw nlwp nsignals nsigs nswap nvcsw nwchan oublk oublock paddr pagein
pcpu pending pgid pid pmem ppid pri re rgid rgroup rss rtprio ruid ruser
sid sig sigcatch sigignore sigmask sl start stat state svgid svuid tdev
time tpgid tsid tsiz tt tty ucomm uid upr uprocp user usrpri vsize vsz
wchan xstat
```

# top command

```
last pid: 52477;  load averages:  0.01,  0.05,  0.02          up 0+19:38:37
17:23:38
29 processes:  1 running, 28 sleeping
CPU states:  0.4% user,  0.0% nice,  0.0% system,  0.0% interrupt, 99.6% idle
Mem: 19M Active, 308M Inact, 113M Wired, 88K Cache, 111M Buf, 556M Free
Swap: 1024M Total, 1024M Free
```

PID	USERNAME	THR	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	COMMAND
697	root	1	76	0	3784K	2728K	select	0:02	0.00%	sshd
565	root	1	76	0	1468K	1068K	select	0:00	0.00%	syslogd
704	root	1	8	0	1484K	1168K	nanslp	0:00	0.00%	cron

## ○ Various usage

- > top -q                   run top and renice it to -20
- > top -u                   don't map uid to username
- > top -Uusername       show process owned by user
- > top -S                 Show system processes in the display

## ○ Interactive command

- > o       change display order (cpu, res, size, time)
- > u       show only processes owned by user (“+” means all)
- > ?       Listing available options

# 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