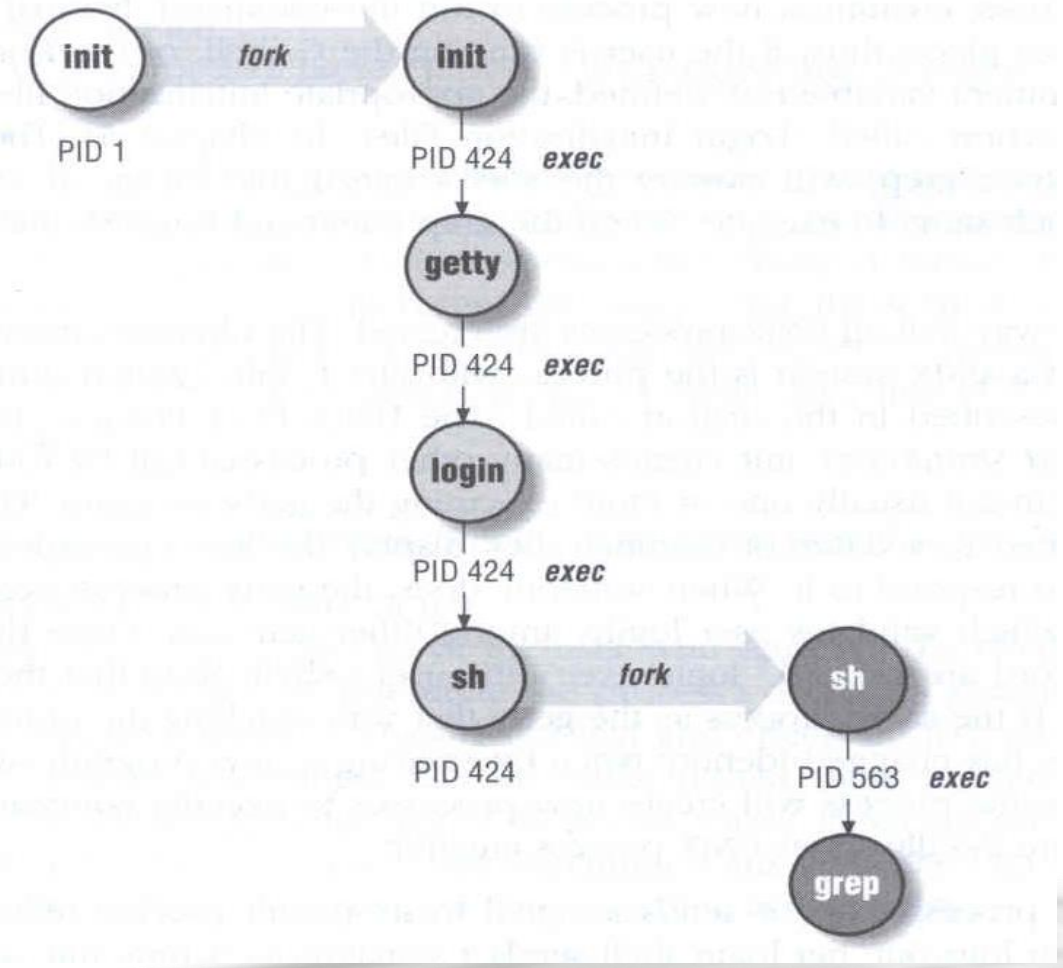
Controlling Process

Computer System and Network Administration

Department of Computer Science & Information Engineering
National Cheng Kung University
2016 Fall

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



Major Attributes of a Process

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

PID and PPID

```
 > cat pid.c
#include <unistd.h>
#include <stdio.h>
int main()
{
        printf("pid = %d, ppid = %d\n", getpid(), getppid());
        return 0;
}
 > ./pid
pid = 66093, ppid = 91200
 > ./pid
pid = 66094, ppid = 91200
 > ./pid
pid = 66095, ppid = 91200
> ps
91200  11  SNs    0:00.09 /bin/tcsh
```

```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;
}
```

```
> ./fork_pid
I am a parent process, my pid is 66544, parent pid is 91200
I am a child process, my pid is 66545, parent pid is 66544
I am a child process, my pid is 66545, parent pid is 66544
I am a parent process, my pid is 66544, parent pid is 91200
...
> I am a child process, my pid is 66545, parent pid is 1
```

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 – _exit(2)
- wait
  - parent use wait to wait for child's death
  - If parent died before child, this orphan process will have init as its new parent – wait(2)

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

```
[/etc] > ls -l *passwd
-rw-------   1 root  wheel      2946 Sep 24 00:26 master.passwd
-rw-r--r--   1 root  wheel      2706 Sep 24 00:26 passwd
[/etc] > ls -l /usr/bin/passwd
-r-sr-xr-x   2 root  wheel      5860 Sep 17 15:19 passwd
```

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>
    - bg, fg
  - by the administrator to achieve various results
    - With kill
  - 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 – FreeBSD signals

- signal(3) or see /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
  - First, find out the pid you want to kill (ps, top, sockstat, lsof…)

  kill [-signal] pid
  - % kill –l (list all available signals)
  - % kill 49222
  - % kill –TERM 49222
  - % kill –15 49222
- killall(1)
  - kill processes by name

  - % killall tcsh
  - % killall -u tsaimh

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

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

```
> ps
  PID  TT  STAT      TIME COMMAND
52363  p0  Ss     0:00.01 -tcsh (tcsh)
52369  p0  R+     0:00.00 ps
```

- ps auxww

```
> ps auxww
USER       PID %CPU %MEM   VSZ   RSS  TT  STAT STARTED      TIME COMMAND
tsaimh   52362  0.0  0.4  6536  3864  ??  S     5:02PM   0:00.02 sshd: tsaimh@ttyp0 (sshd)
root     52380  0.0  0.3  3756  3224  ??  Ss    5:08PM   0:00.00 sendmail: accepting connections (sendmail)
smmsp    52384  0.0  0.3  3644  2968  ??  Ss    5:08PM   0:00.00 sendmail: Queue runner@00:30:00 for
/var/spool/clientmqueue (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

Use these options with shell scripts

```
> ps -j
USER     PID  PPID  PGID   SID JOBC STAT  TT       TIME COMMAND
tsaimh 52363 52362 52363 52363    0 Ss    p0    0:00.03 -tcsh (tcsh)
tsaimh 52458 52363 52458 52363    1 R+    p0    0:00.00 ps -j
```

- ps -o

```
> ps -o uid,pid,ppid,%cpu,%mem,command
  UID   PID  PPID %CPU %MEM COMMAND
 1001 52363 52362  0.0  0.3 -tcsh (tcsh)
 1001 52462 52363  0.0  0.1 ps -o uid,pid,ppid,%cpu,%mem,command
```

- ps -L

```
> ps -L
%cpu %mem acflag acflg args blocked caught comm command cpu cputime emuletime 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: 67104;  load averages:  3.08,  3.02,  3.01
up 72+03:43:31  19:46:49
131 processes: 4 running, 125 sleeping, 2 stopped
CPU: 37.5% user,  0.0% nice,  0.0% system,  0.0% interrupt, 62.5% idle
Mem: 224M Active, 2889M Inact, 223M Wired, 94M Cache, 112M Buf, 77M Free
Swap: 4096M Total, 8460K Used, 4088M Free

  PID USERNAME    THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
59857 waters        1 118    0  2776K  1140K CPU6    6  29.8H 99.27% 47Throughput
98836 waters        1 116    0  3360K   948K CPU7    7 294.2H 99.07% 400Infinty
65306 waters        1 117    0  2752K  1236K CPU1    1 525:04 93.26% 32Throughput
67102 root          1  46    0  3788K  1856K sbwait  0   0:01  1.76% ftpd
```

- 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 (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 –TERM for unknown process
  - renice it to a higher nice value for reasonable process