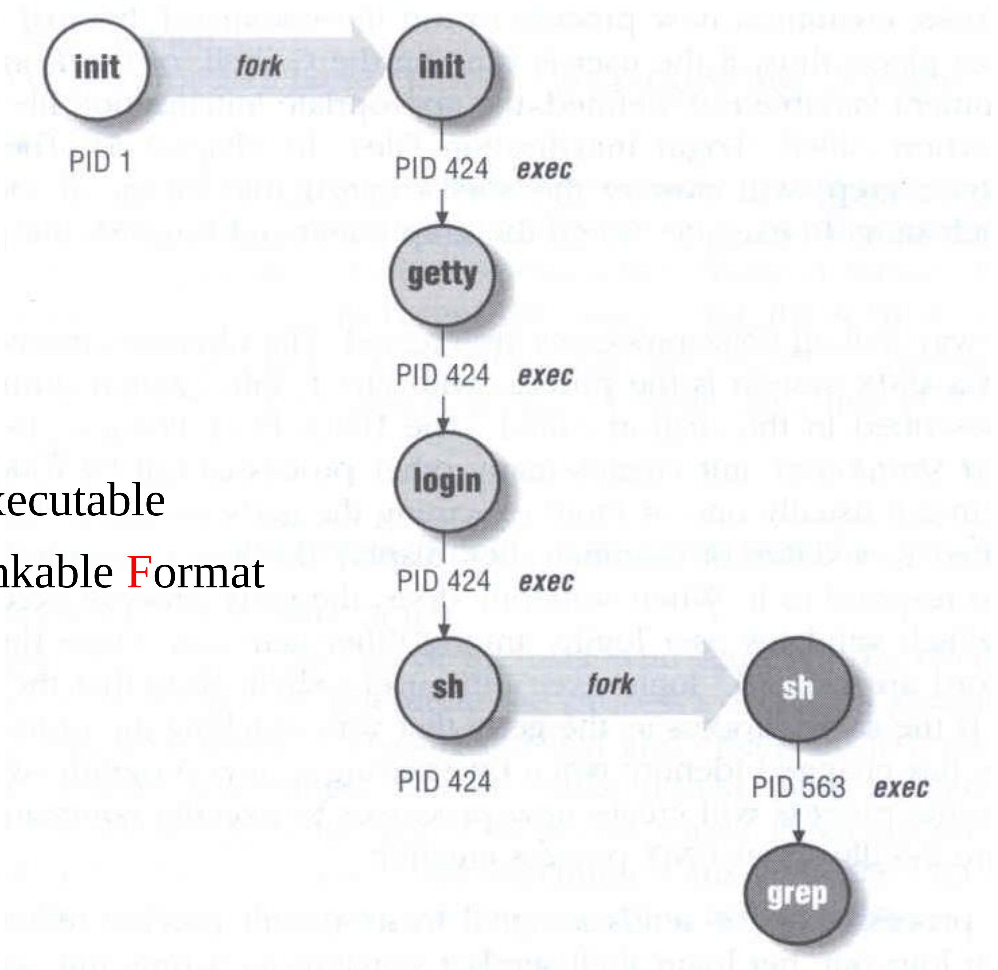
Controlling Processes

lctseng / Liang-Chi Tseng

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
      - Executable and Linkable Format
- When you execute it
  - It becomes a process
- Process is alive
  - It resides in memory



Components of a Process

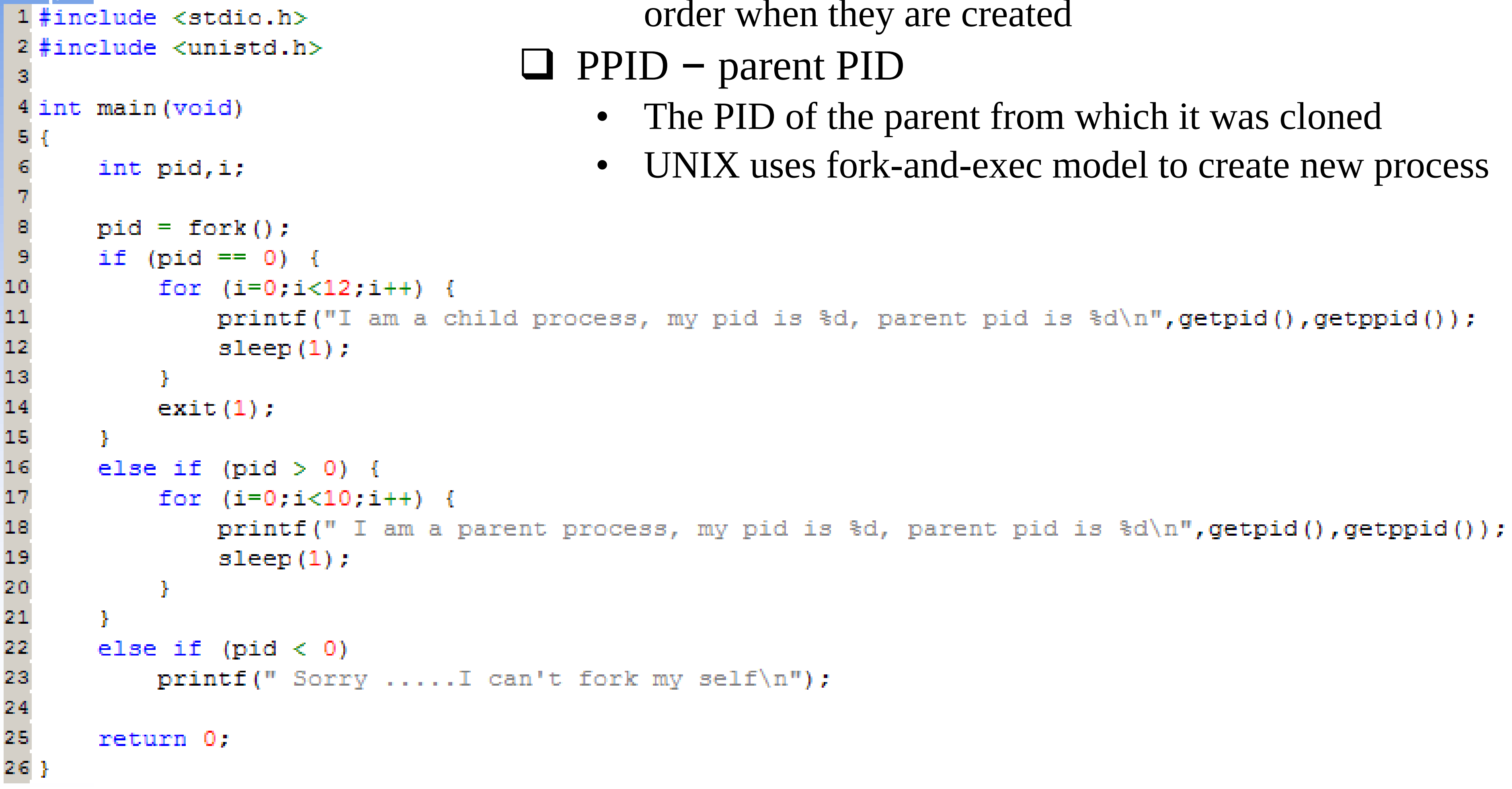
- An address space in memory
  - Code and data of this process
- A set of data structures within the kernel
  - Used to monitor, schedule, trace, …., this process
    - Owner, Group (Credentials)
    - Current status
    - VM space
    - Execution priority (scheduling info)
    - Information of used resource
    - Resource limits
    - Syscall vector
    - Signal actions

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;
}
```

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 it's new parent – wait(2)

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

Important!!

- 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

```
sabsd [/etc] -lctseng- ls -al | grep passwd
-rw-------   1 root  wheel      2946 Sep 24 00:26 master.passwd
-rw-r--r--   1 root  wheel      2706 Sep 24 00:26 passwd
sabsd [/usr/bin] -lctseng- ls -al /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
    - divide by zero
    - Illegal memory access

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

FreeBSD

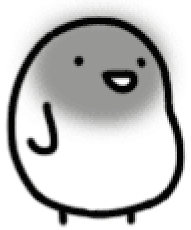
#	Name	Description	Default	Catch	Block	Dump core
1	SIGHUP	Hangup	Terminate	✓	✓	⊘
2	SIGINT	Interrupt (^C)	Terminate	✓	✓	⊘
3	SIGQUIT	Quit	Terminate	✓	✓	✓
9	SIGKILL	Kill	Terminate	⊘	⊘	⊘
10	SIGBUS	Bus error	Terminate	✓	✓	✓
11	SIGSEGV	Segmentation fault	Terminate	✓	✓	✓
15	SIGTERM	Soft. termination	Terminate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

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
    - $ killall tcsh
    - $ killall -u lctseng

Niceness

- How kindly of you when contending CPU time
  - High nice value ➔ low priority
  - Related to CPU time quantum
- 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(built-in) : $ nice [range] utility [argument]
    - $ nice +10 ps -l
- renice format
  - % renice [prio | -n incr] [-p pid] [-g gid] [-u user]
    - % renice 15 -u lctseng

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

Niceness

- Example relation between niceness and CPU time quantum

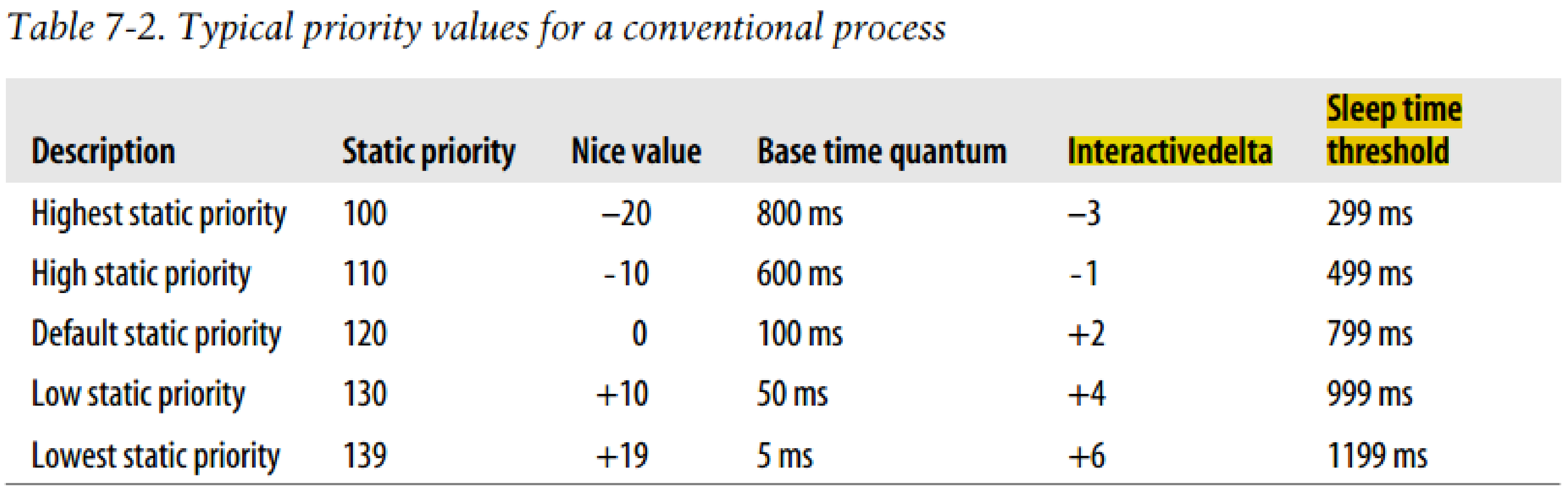


Table 7-2. Typical priority values for a conventional process

Description	Static priority	Nice value	Base time quantum	Interactivedelta	Sleep time threshold
Highest static priority	100	–20	800 ms	–3	299 ms
High static priority	110	-10	600 ms	-1	499 ms
Default static priority	120	0	100 ms	+2	799 ms
Low static priority	130	+10	50 ms	+4	999 ms
Lowest static priority	139	+19	5 ms	+6	1199 ms

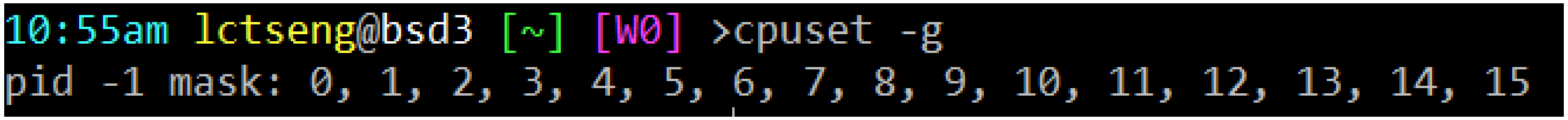
Cited from NCTU-OSDI slides

cpuset command

- A system may have more than one CPU core
- How many CPU resource a process can use
- cpuset(1)

cpuset command

- To see how many CPUs on your machine
  - cpuset -g



```
10:55am lctseng@bsd3 [~] [W0] >cpuset -g
pid -1 mask: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
```

- Run commands with less CPUs
  - cpuset -l *cpus cmd*
  - cpuset -l 8-15 ./hw1.out
- Change number of CPUs for current processes
  - cpuset -l *cpus* -p *pid*
  - cpuset -l 8-15 -p 5566
- Combine with nice
  - cpuset -l 8-15 /usr/bin/nice -n 20 *cmd*

Process States

- man “ps” and see “state” keyword

State	Meaning
I	Idle (20+ second)
R	Runnable
S	Sleeping (~20 second)
T	Stopped
Z	Zombie
D	in Disk

ps command (BSD、Linux)

- ps

```
sabsd [/home/lctseng] -lctseng- ps
  PID  TT  STAT      TIME COMMAND
52363  p0  Ss     0:00.01 -tcsh (tcsh)
52369  p0  R+     0:00.00 ps
```

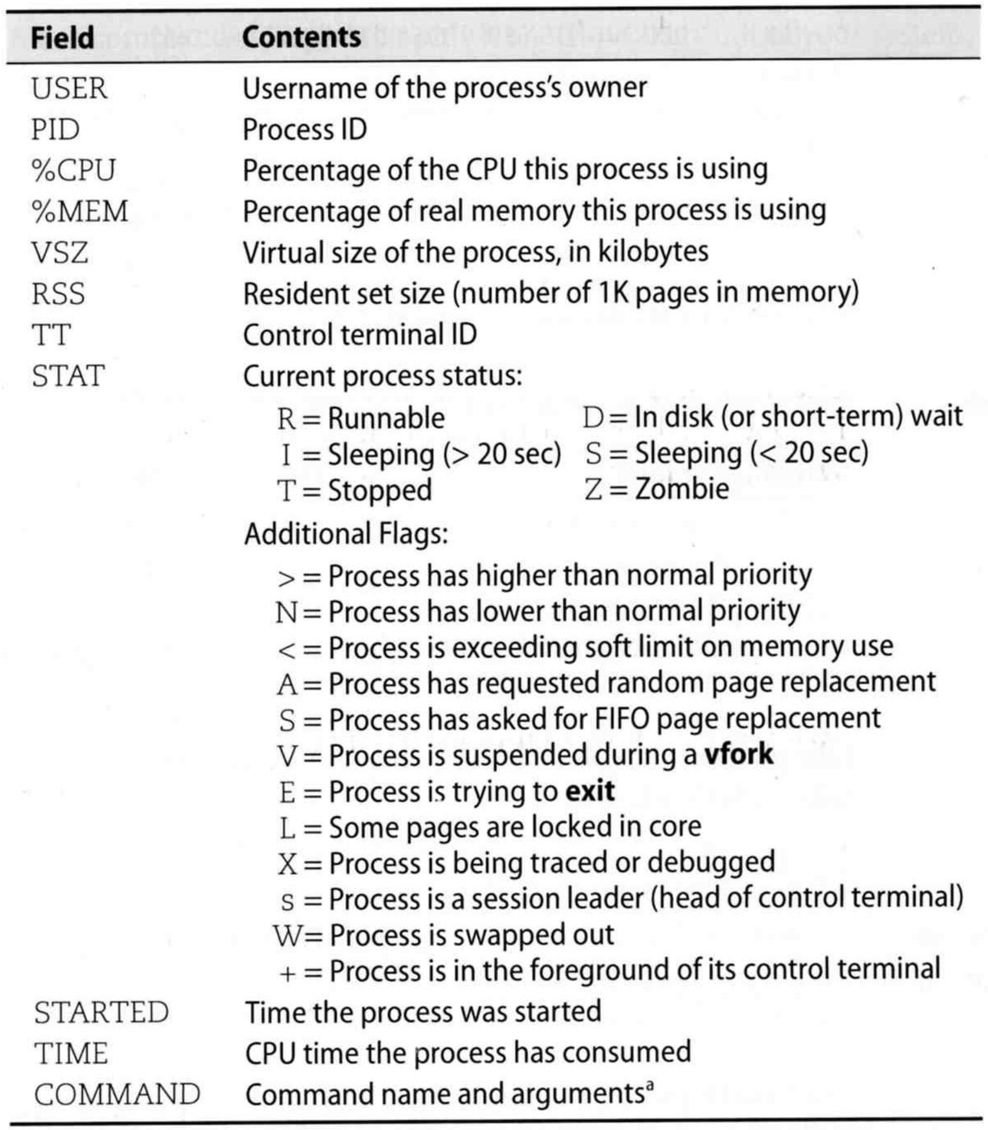
- ps aux

```
sabsd [/home/lctseng] -lctseng- ps aux
USER       PID %CPU %MEM   VSZ   RSS  TT  STAT STARTED      TIME COMMAND
lctseng   52362  0.0  0.4  6536  3852  ??  S      5:02PM   0:00.01 sshd: lctseng@ttyp0 (sshd)
root     52380  0.0  0.3  3756  3224  ??  Ss     5:08PM   0:00.00 sendmail: accepting connections (s
smmsp    52384  0.0  0.3  3644  2968  ??  Ss     5:08PM   0:00.00 sendmail: Queue runner@00:30:00 fo
```

- ps auxww

```
sabsd [/home/lctseng] -lctseng- ps auxww
USER       PID %CPU %MEM   VSZ   RSS  TT  STAT STARTED      TIME COMMAND
lctseng   52362  0.0  0.4  6536  3864  ??  S      5:02PM   0:00.02 sshd: lctseng@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)

Use these options with shell scripts

- ps -j

```
sabsd [/home/lctseng] -lctseng- ps -j
USER      PID  PPID  PGID   SID JOBC STAT  TT       TIME COMMAND
lctseng 52363 52362 52363 52363    0 Ss     p0    0:00.03 -tcsh (tcsh)
lctseng 52458 52363 52458 52363    1 R+     p0    0:00.00 ps -j
```

- ps –o

```
sabsd [/home/lctseng] -lctseng- 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

```
sabsd [/home/lctseng] -lctseng- 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: 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
```

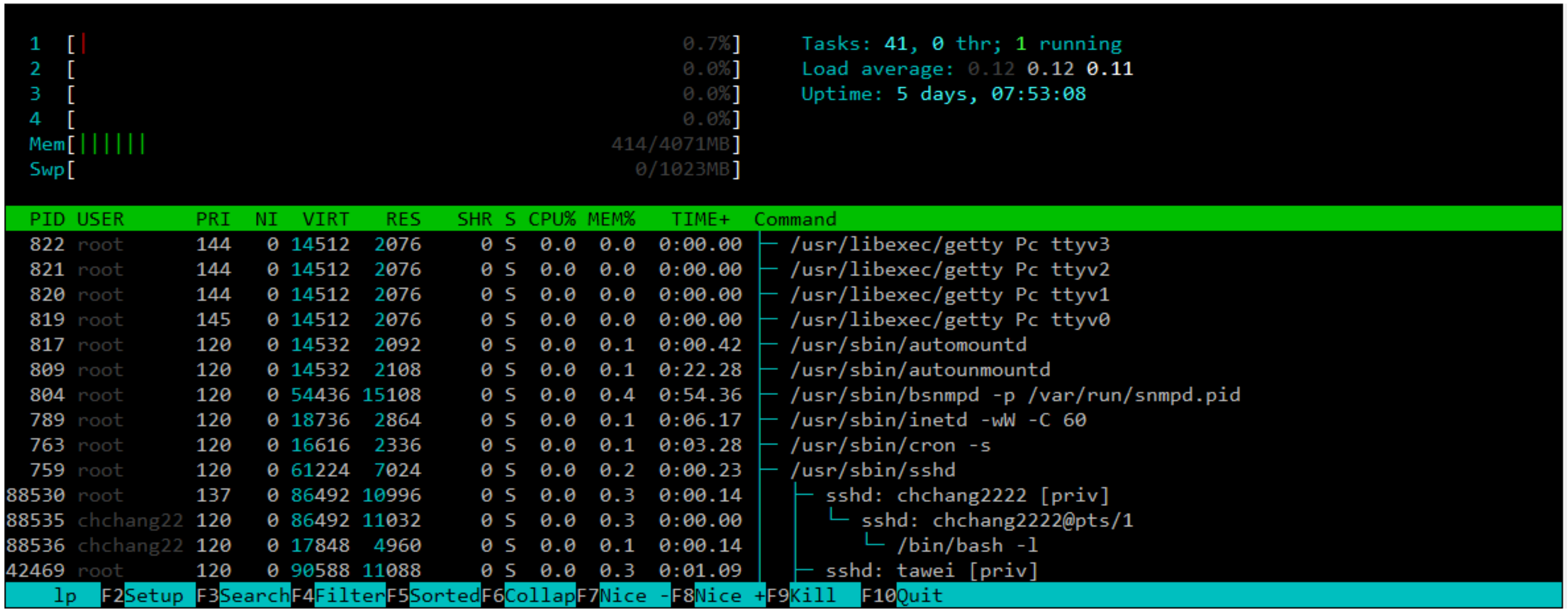
- Various usage

• top -q	run top and renice it to -20
• top -u	don’t map uid to username
• top -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)
• m	show IO information
• ?	Listing available options

htop command



```
  1  [|                                                   0.7%]     Tasks: 41, 0 thr; 1 running
  2  [                                                    0.0%]     Load average: 0.12 0.12 0.11
  3  [                                                    0.0%]     Uptime: 5 days, 07:53:08
  4  [                                                    0.0%]
  Mem[||||||                                       414/4071MB]
  Swp[                                               0/1023MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
  822 root      144   0 14512  2076     0 S  0.0  0.0  0:00.00 ├─ /usr/libexec/getty Pc ttyv3
  821 root      144   0 14512  2076     0 S  0.0  0.0  0:00.00 ├─ /usr/libexec/getty Pc ttyv2
  820 root      144   0 14512  2076     0 S  0.0  0.0  0:00.00 ├─ /usr/libexec/getty Pc ttyv1
  819 root      145   0 14512  2076     0 S  0.0  0.0  0:00.00 ├─ /usr/libexec/getty Pc ttyv0
  817 root      120   0 14532  2092     0 S  0.0  0.1  0:00.42 ├─ /usr/sbin/automountd
  809 root      120   0 14532  2108     0 S  0.0  0.1  0:22.28 ├─ /usr/sbin/autounmountd
  804 root      120   0 54436 15108     0 S  0.0  0.4  0:54.36 ├─ /usr/sbin/bsnmpd -p /var/run/snmpd.pid
  789 root      120   0 18736  2864     0 S  0.0  0.1  0:06.17 ├─ /usr/sbin/inetd -wW -C 60
  763 root      120   0 16616  2336     0 S  0.0  0.1  0:03.28 ├─ /usr/sbin/cron -s
  759 root      120   0 61224  7024     0 S  0.0  0.2  0:00.23 ├─ /usr/sbin/sshd
88530 root      137   0 86492 10996     0 S  0.0  0.3  0:00.14 │  ├─ sshd: chchang2222 [priv]
88535 chchang22 120   0 86492 11032     0 S  0.0  0.3  0:00.00 │  │  └─ sshd: chchang2222@pts/1
88536 chchang22 120   0 17848  4960     0 S  0.0  0.1  0:00.14 │  │     └─ /bin/bash -l
42469 root      120   0 90588 11088     0 S  0.0  0.3  0:01.09 │  ├─ sshd: tawei [priv]
   lp  F2Setup F3SearchF4FilterF5SortedF6CollapF7Nice -F8Nice +F9Kill  F10Quit
```

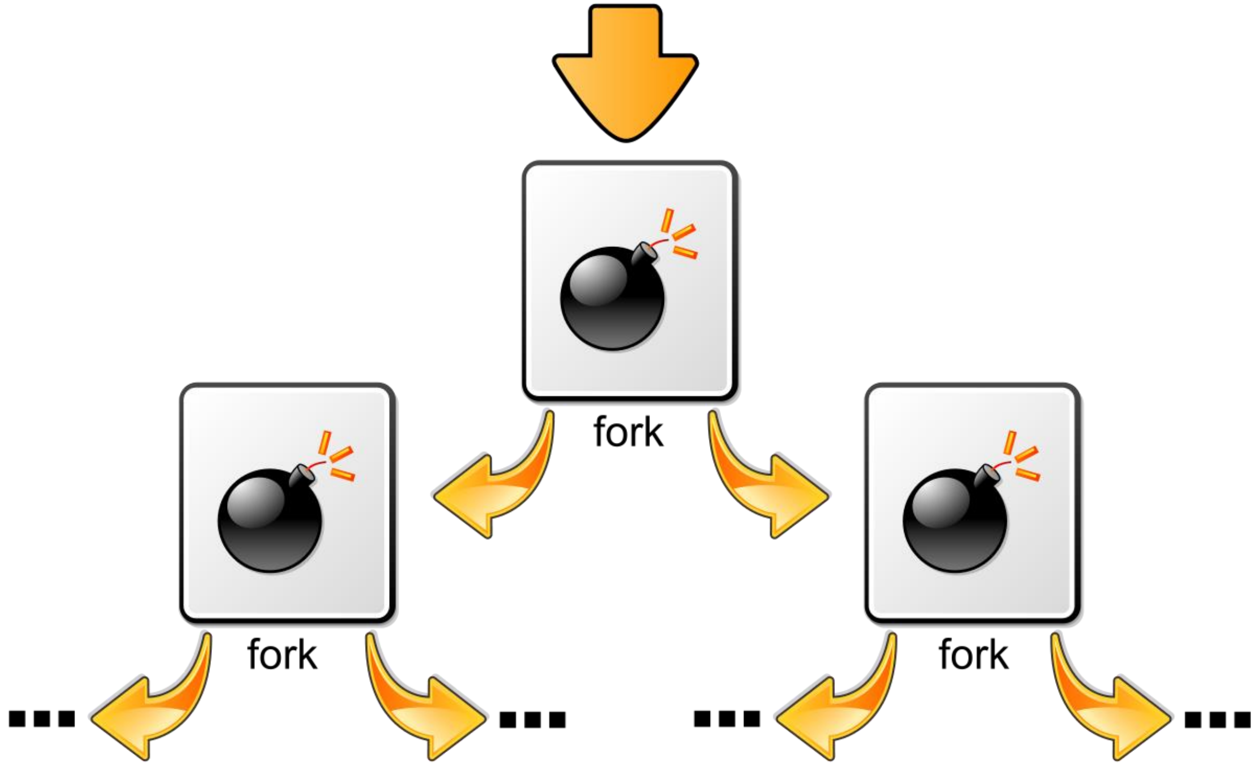
- A better top
  - Install it from sysutils/htop

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

Fork Bomb

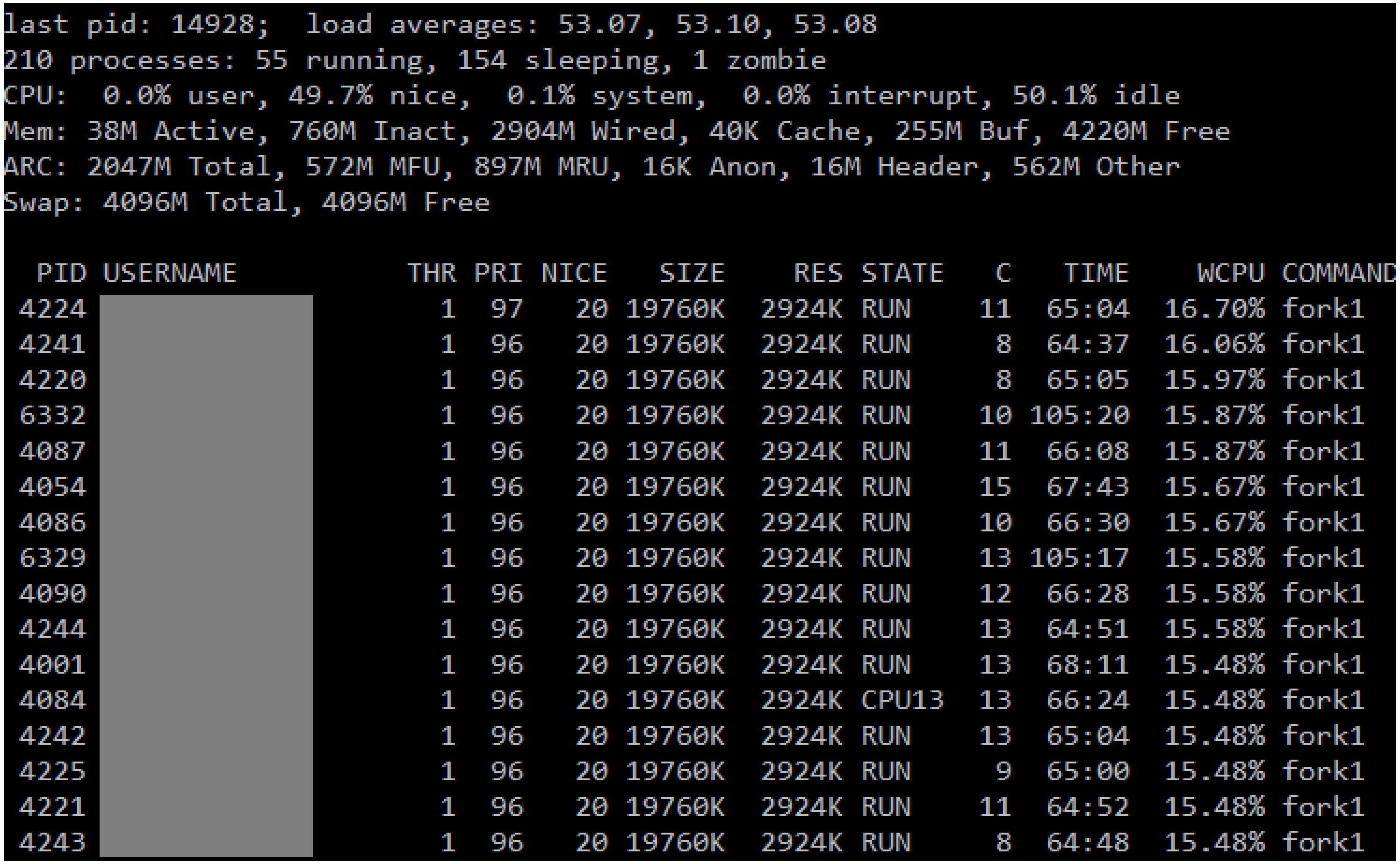
- A process forking out of control



Cited from wiki

Fork Bomb

❑ A process forking out of control



```
last pid: 14928;  load averages: 53.07, 53.10, 53.08
210 processes: 55 running, 154 sleeping, 1 zombie
CPU:  0.0% user, 49.7% nice,  0.1% system,  0.0% interrupt, 50.1% idle
Mem: 38M Active, 760M Inact, 2904M Wired, 40K Cache, 255M Buf, 4220M Free
ARC: 2047M Total, 572M MFU, 897M MRU, 16K Anon, 16M Header, 562M Other
Swap: 4096M Total, 4096M Free

  PID USERNAME       THR PRI NICE   SIZE    RES STATE   C   TIME    WCPU COMMAND
 4224 [redacted]       1  97   20 19760K  2924K RUN    11  65:04  16.70% fork1
 4241                  1  96   20 19760K  2924K RUN     8  64:37  16.06% fork1
 4220                  1  96   20 19760K  2924K RUN     8  65:05  15.97% fork1
 6332                  1  96   20 19760K  2924K RUN    10 105:20  15.87% fork1
 4087                  1  96   20 19760K  2924K RUN    11  66:08  15.87% fork1
 4054                  1  96   20 19760K  2924K RUN    15  67:43  15.67% fork1
 4086                  1  96   20 19760K  2924K RUN    10  66:30  15.67% fork1
 6329                  1  96   20 19760K  2924K RUN    13 105:17  15.58% fork1
 4090                  1  96   20 19760K  2924K RUN    12  66:28  15.58% fork1
 4244                  1  96   20 19760K  2924K RUN    13  64:51  15.58% fork1
 4001                  1  96   20 19760K  2924K RUN    13  68:11  15.48% fork1
 4084                  1  96   20 19760K  2924K CPU13  13  66:24  15.48% fork1
 4242                  1  96   20 19760K  2924K RUN    13  65:04  15.48% fork1
 4225                  1  96   20 19760K  2924K RUN     9  65:00  15.48% fork1
 4221                  1  96   20 19760K  2924K RUN    11  64:52  15.48% fork1
 4243                  1  96   20 19760K  2924K RUN     8  64:48  15.48% fork1
```

Fork Bomb – How to create a fork bomb

- C/C++

```c
#include <unistd.h>

int main()
{
  while(1)
    fork();
  return 0;
}
```

- Bash (Shell script)

```bash
:(){ :|:& };:
```

```bash
# 定義函式
forkbomb()
{
    # 使用pipe呼叫兩次，並丟到背景執行
    forkbomb|forkbomb &
}
;
# 執行函式，引爆fork bomb
forkbomb
```

- Perl

```perl
fork while fork
```

- Windows

```
%0|%0
```

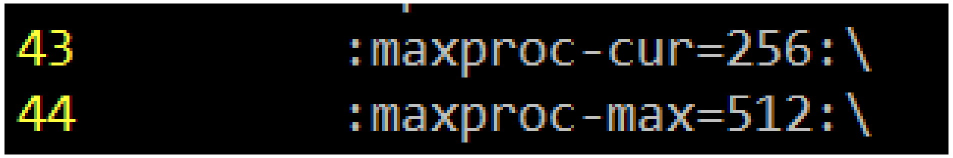
DON’T DO THAT!!!!

Fork Bomb

- How to deal with fork bomb
  - Just kill all of them
  - `$ killall -KILL bombName`
- When you have no more resource to fork you shell
  - `$ exec killall -KILL bombName`
  - That shell will become 'killall', and never goes back
- 'killall' isn't an atomic command
  - More bombs may be created when killing them
  - Run multiple 'killall'

Fork Bomb

- ❑ Prevent fork bomb
  - Limit the maximum number of processes for a specific user
- ❑ /etc/login.conf



```
                :maxproc-cur=256:\
                :maxproc-max=512:\
```