

Linux Systems and Open Source Software

Process Management

Chia-Heng Tu
Dept. of Computer Science and Information Engineering
National Cheng Kung University
Fall 2022



Outline

- CPU Resource Management
- Processes in Linux
- Job Control
- Process Management
  - Process observation
  - Process management
  - System resource management



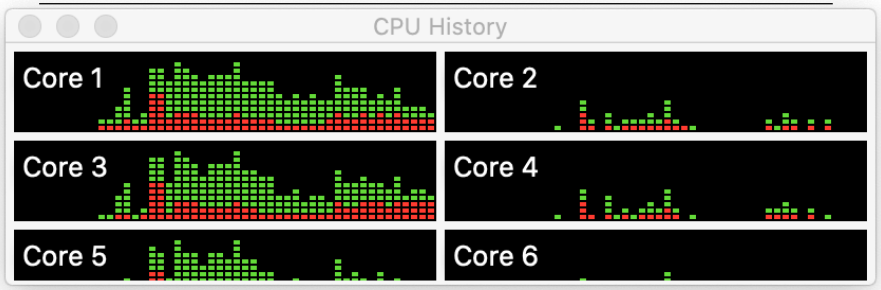
CPU RESOURCE MANAGEMENT



CPU Resource Management

- Why you want to manage your CPU resource?
  - You want to run a heavy loading program, e.g., games
  - You want your 4 cores CPU to do 8 or more tasks in a time
  - You want to enhance your software performance
  - Just want to know and master your computer!

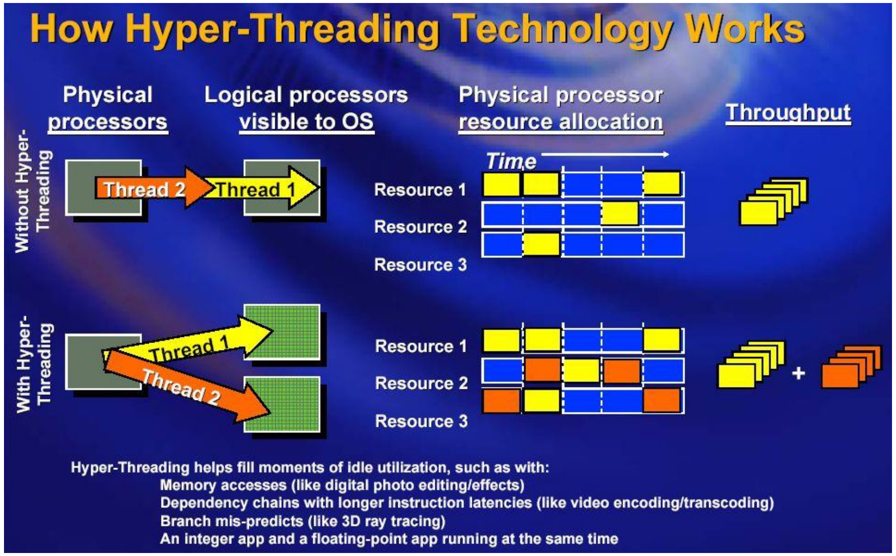
CPU Utilization of a Multicore Processor



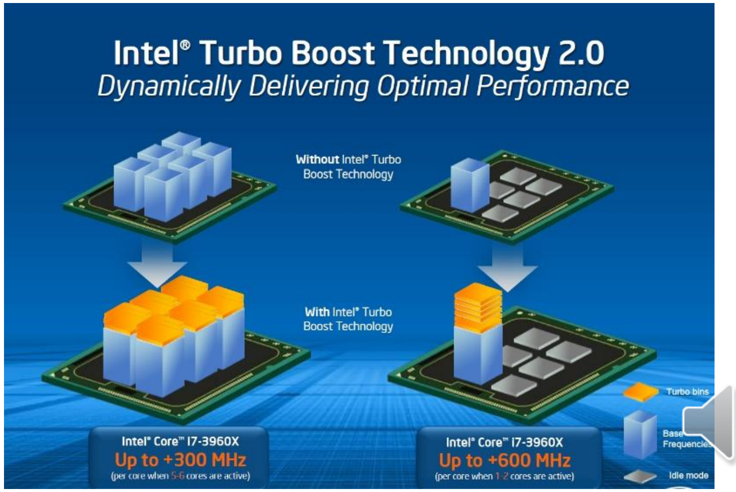
Courtesy of https://en.wikipedia.org/wiki/Hyper-threading; How to check Mac CPU usage

Modern CPU Technologies

- Intel Hyper-Threading(HT) Technology
  - Intel's proprietary simultaneous multithreading (SMT) implementation permitting multiple independent threads of execution to better utilize the resources
  - (OS perspective) For each physical core, the operating system addresses two virtual (logical) cores and shares the workload
  - (HW perspective) One physical core equips with two sets of registers



- Intel Turbo Boost
  - Intel Turbo Boost raises its processors' operating frequency (i.e., overclocking), when demanding tasks are running
  - It dynamically boosts the CPU frequencies according to the number of activating cores



- You might like to know more about the related technologies
  - NUMA (Non-uniform memory access)
  - SMP (Symmetric Multi-Processing)

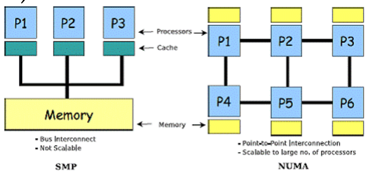


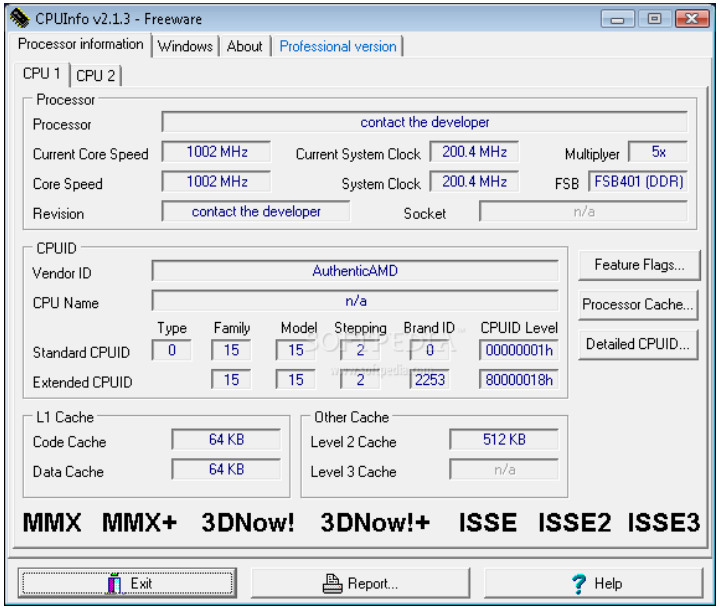
Figure 1: SMP versus NUMA

Courtesy of https://en.wikipedia.org/wiki/Intel_Turbo_Boost; Critical Flaw in HyperThreading Discovered in "Skylake" and "Kaby Lake" CPUs; Intel Second Generation Core i7 3820 Review; MultiCore Systems: An Overview

List CPU Information

- $cat /proc/cpuinfo
  - **processor** – provides each processor with an identifying number. If you have eight processors it will display eight times from 0-7
  - **cpu family** – Authoritatively tells you the type of processor you have in the system
  - **model name** – Gives you the common name of the processor, including the project name
  - **cpu MHz** – Shows the processor's precise speed, in megahertz, to the thousandth decimal point
  - **cache size** – Tells you the amount of level 2 memory cache available to the processor
  - **flags** – Defines a number of different processor attributes, such as the presence of a floating-point unit (FPU) and the ability to process MMX instructions

CPU Status Software in Windows



CPU information in Linux

```
$ more /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz
stepping        : 9
microcode       : 0x12
cpu MHz         : 1200.000
cache size      : 6144 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm c
onstant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf
 eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm p
cid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lah
f_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips        : 4190.56
clflush size    : 64
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
power management:
```



CPU Affinity

- **numactl** - control NUMA policy for processes or shared memory

```
[root@study ~]# numactl [-C CPU_id] command
選項與參數：
-C：輸入想要運行的 CPU id
範例一：將 top 放到 CPU 5 號運行。
[root@study ~]# numactl -C 5 top
...top 運作視窗...

# 開啟另外一個 terminal 用 ps 來檢查，PSR 就是 process 運行的核心
[root@study ~]# ps –o pid,psr,comm –p 14790
  PID PSR COMMAND
14790   5 top
```

numastat to check NUMA status

- **taskset** - set or retrieve a process's CPU affinity (w/o memory affinity)

```
[root@study ~]# taskset [-c mask] [-p pid] command
選項與參數：
-c：輸入想要運行的 CPU id
-p：想要改變 CPU addinity 的 PID
範例一：將 pid 19767 放到 CPU 1 號運行。
[root@localhost ~]# taskset -cp 1 19767
pid 19767's current affinity list: 0  <== 將 0 號改成 1 號 CPU 去執行囉！
pid 19767's new affinity list: 1
```

Courtesy of numactl(8) - Linux man page; taskset(1) - Linux man page
3.3. 配置建議 Red Hat Enterprise Linux 7 | Red Hat Customer Portal



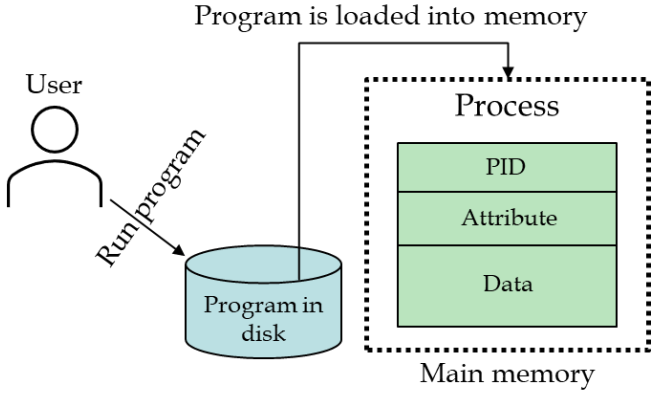
PROCESSES IN LINUX



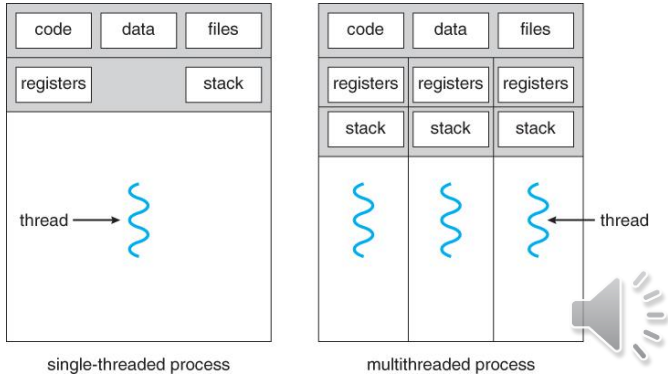
Processes in Linux

- Program vs. Process
  - A **Program** is a file accommodating a group of instructions to be executed (i.e., executable file)
  - A **Process/Thread** is **an execution instance** of a program
    - Multiple processes can be related to the same program
  - Operating System handles processes via **PCB** (Process Control Block)
    - which includes the contents of program counter, stack, state, etc., required during program execution

Load program into memory for execution



A thread is a basic execution unit on a processor core



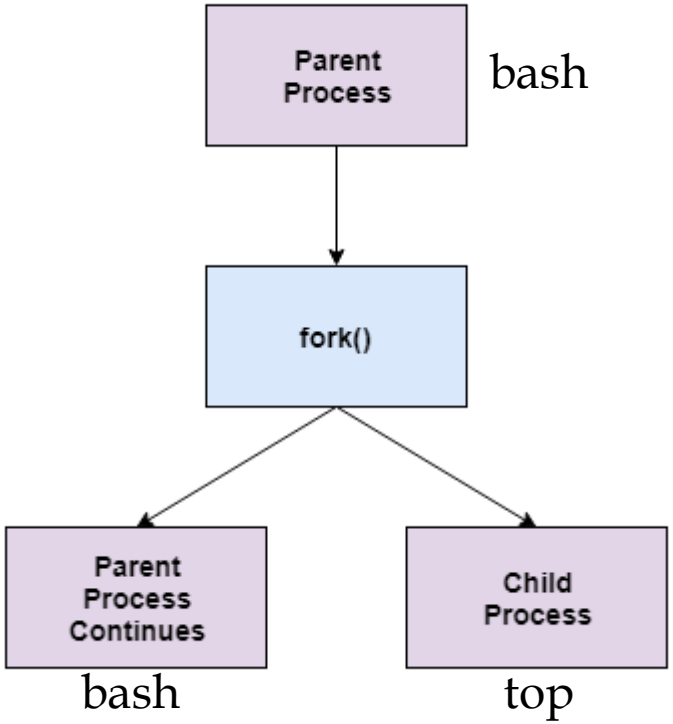
Courtesy of https://techdifferences.com/difference-between-program-and-process.html; Threads



Parent and Child Processes

- A **child** process is a process created by a **parent** process
  - by operating system using a **fork()** system call

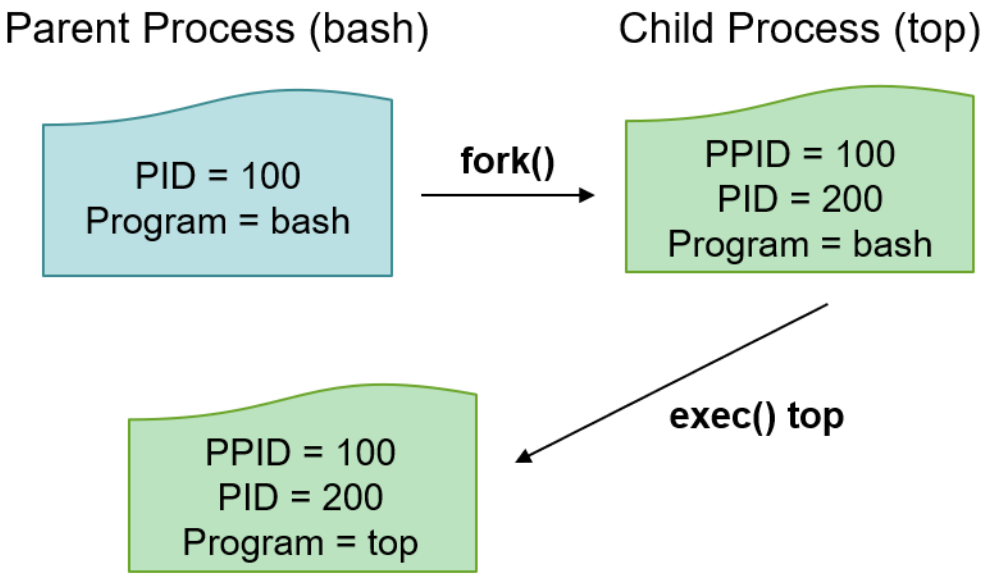
- The child process will record its parent PID (process identifier) with PPID
  - E.g., when you use **bash** to execute a command **top**, **top** is a child process and **bash** is the parent process





fork-and-exec in Linux

- **fork()** starts a child process
  - which is a copy of the parent process that calls it,
  - while **exec()** replaces the current process image with another (different) one





JOB CONTROL



Multiuser, Multitasking Operating System

- **Multiuser OS** allows multiple people to use a computer and they do not affect each other's stuff (files, preferences, etc.)
  - In Linux, multiple users can even use the computer simultaneously
  - The **~/.bashrc** file records each user's preference
- **Multitasking OS** allows more than one program (called a "task") to run at the same time and share the system resources

- Linux has seven default terminals
  - One for GUI and six for CLIs (command line interfaces)
  - Use short key **[ALT]+[F1]-[F7]** to switch the terminal
  - E.g., if process A causes a deadlock in terminal 1, you can switch to terminal 2 to kill process A



What is Job Control?

- A job is a process that the shell manages
  - Each job is assigned a sequential job ID and an associated PID
- There are three types of job statuses:
  - **Foreground**: When you enter a command, the command occupies that terminal window until it completes. This is a foreground job
  - **Background**: When you enter an ampersand (**&**) symbol at the end of a command line, the command runs without occupying the terminal window. This is an example of a background job
  - **Stopped**: If you press **Control + 'Z'** for a foreground job, or enter the stop command for a background job, the job stops. This job is called a stopped job



Job Control Commands

- **&** - let command executes in the background
  - "**&**" will return **[job id]** and **PID** for user to track the background jobs
  - The jobs' output will still print on the terminal, but it won't occupy it
  - To hide the output, you can use the ***redirection***, e.g., '>' or '>>'

```
[root@study ~]# tar -zpcf /tmp/etc.tar.gz /etc &
[1] 14432  <==  [job number] PID
[root@study ~]# tar: Removing leading `/' from member names
[1]+  Done                    tar -zpcf /tmp/etc.tar.gz /etc  <==  task finish
```

- **[ctrl] + 'z'** - stops the foreground job and places it in the background as a stopped job

```
[root@study ~]# vim ~/.bashrc
# in vim normal mode press [ctrl]-z
[1]+ Stopped vim ~/.bashrc
[root@study ~]# <== stop the job and get the foreground control.
```



Job Control Commands (Cont'd)

- **jobs** - lists all background jobs

```
[root@study ~]# jobs [-lrs]
選項與參數：
-l ：除了列出 job number 與指令串之外，同時列出 PID 的號碼；
-r ：僅列出正在背景 run 的工作；
-s ：僅列出正在背景當中暫停 (stop) 的工作。
範例一：觀察目前的 bash 當中，所有的工作，與對應的 PID
[root@study ~]# jobs –l
[1]- 14566 Stopped vim ~/.bashrc
```

- **fg** - brings the current or specified job into the foreground
- **bg** - places the current or specified job in the background

```
範例一：先以 jobs 觀察，再將工作2取出：
[root@study ~]# jobs –l
[1]- 14566 Stopped vim ~/.bashrc
[2]+ 14567 Stopped find / -print
[root@study ~]# fg %2  '%' 是用來指定job的ID
範例二：讓停止的工作1在背景下運行：
[root@study ~]# bg %1 ; jobs
[1]- Stopped find / -print
[1] Running find / -print
```



Job Control Commands (Cont'd)

- **kill** - terminate a process

```
[root@study ~]# kill -signal %jobnumber
[root@study ~]# kill -l
選項與參數：
-l ：列出所有 signal。
-2 ：代表與由鍵盤輸入 [ctrl]-c 同樣的動作；
-9 ：立刻強制刪除一個工作；
-15：以正常的程序方式終止一項工作。與 -9 是不一樣的。

範例一：找出目前的 bash 環境下的背景工作，並將該工作『強制刪除』。
[root@study ~]# jobs
[1]+ Stopped vim ~/.bashrc
[2] Stopped find / -print
[root@study ~]# kill -9 %2; jobs
[1]+ Stopped vim ~/.bashrc
[2] Killed find / -print

範例二：找出目前的 bash 環境下的背景工作，並將該工作『正常終止』掉。
[root@study ~]# jobs
[1]+ Stopped vim ~/.bashrc
[root@study ~]# kill -SIGTERM %1
# -SIGTERM 與 -15 是一樣的，可以使用 kill -l 來查閱。
# 不過在這個案例中， vim 的工作無法被結束，因為其無法透過 kill 正常終止。
```



Process status report

Process management

System resource monitor

PROCESS MANAGEMENT



Process Management in Linux

- Linux creates a process whenever a program is launched
  - A process is a container of information about how that program is running and what's happening
- Here are the things you may want to do when managing Linux processes:
  - **See** the running processes
  - See how many processes your Linux system are running (especially any greedy ones)
  - **Locate** a particular process to see what it's doing or to do something w/ it
  - Define or change the level of **priority** associated with that process
  - **Terminate** the process if it has outlived its usefulness or if it's misbehaving



Process Status Report Command - ps

- **ps** - report a snapshot of the current processes
  - **F:** process flags
    - **1** forked but didn't exec
    - **4** used super-user privileges
  - **S:** process state codes
    - **R** running or runnable (on run queue)
    - **S** interruptible sleep (waiting for an event to complete)
    - **T** stopped by job control signal
    - **Z** defunct ("zombie") process, terminated but not reaped by its parent

```
[root@study ~]# ps -l
F S UID   PID  PPID C PRI NI ADDR SZ WCHAN  TTY      TIME CMD
4 S   0 14830 13970 0  80   0 - 52686 poll_s pts/0 00:00:00 sudo
4 S   0 14835 14830 0  80   0 - 50511 wait   pts/0 00:00:00 su
4 S   0 14836 14835 0  80   0 - 29035 wait   pts/0 00:00:00 bash
0 R   0 15011 14836 0  80   0 - 30319 -      pts/0 00:00:00 ps
# ps -l 預設會以當前的 bash 為父程序(parent) 去顯示所有子程序(child)。
```



Process Observation Command - ps (Cont'd)

- **ps** - report a snapshot of the current processes

```
範例四：列出類似程序樹的程序顯示，亦可使用pstree指令
[root@study ~]# ps axjf
 PPID   PID  PGID   SID TTY   TPGID STAT UID TIME COMMAND
    0     2     0     0 ?       -1 S      0 0:00 [kthreadd]
    2     3     0     0 ?       -1 S      0 0:00  \_ [ksoftirqd/0]
.....(中間省略).....
    1  1326  1326  1326 ?       -1 Ss     0 0:00 /usr/sbin/sshd –D
 1326 13923 13923 13923 ?       -1 Ss     0 0:00  \_ sshd: dmtsai [priv]
13923 13927 13923 13923 ?       -1 S   1000 0:00     \_ sshd: dmtsai@pts/0
13927 13928 13928 13928 pts/0 18703 Ss  1000 0:00       \_ -bash
13928 13970 13970 13928 pts/0 18703 S   1000 0:00         \_ bash
13970 14830 14830 13928 pts/0 18703 S      0 0:00           \_ sudo su –
14830 14835 14830 13928 pts/0 18703 S      0 0:00             \_ su –
14835 14836 14836 13928 pts/0 18703 S      0 0:00               \_ -bash
14836 18703 18703 13928 pts/0 18703 R+     0 0:00                 \_ ps axjf

範例五：找出與 cron 與 rsyslog 這兩個服務有關的 PID 號碼
[root@study ~]# ps aux | egrep '(cron|rsyslog)' 省找出系統所有processes，篩出名稱為cron 或 rsyslog字串的程序
root   742 0.0 0.1 208012 4088 ?     Ssl Aug04 0:00 /usr/sbin/rsyslogd –n
root  1338 0.0 0.0 126304 1704 ?     Ss  Aug04 0:00 /usr/sbin/crond –n
root 18740 0.0 0.0 112644  980 pts/0 S+  00:49 0:00 grep -E --color=auto (cron|rsyslog)
```



Process Observation Command - pstree

- **pstree** - shows running processes as a tree

```
[root@study ~]# pstree [-A|U] [-up]
選項與參數：
-p ：並同時列出每個 process 的 PID；
-u ：並同時列出每個 process 的所屬帳號名稱。

[root@study ~]# pstree –up
systemd(1)-+-ModemManager(745)-+-{ModemManager}(785)
           |                   `-{ModemManager}(790)
           |-NetworkManager(870)-+-{NetworkManager}(907)
           |                     |-{NetworkManager}(911)
           |                     `-{NetworkManager}(914)
....(中間省略)....
           |-sshd(1326)---sshd(13923)---sshd(13927,dmtsai)---bash(13928)---bash(13970)---
....(底下省略)....
# 在括號 () 內的即是 PID 以及該程序的 owner 喔！一般來說，如果該程序的所有人與父程序同，
# 就不會列出，但是如果與父程序不一樣，那就會列出該程序的擁有者！看上面 13927 就轉變成 dmtsai 了
```



Process Observation Command - top

- **top** - the **top** program provides a dynamic real-time view of a running system

```
[root@study ~]# top [-d 數字] | top [-bnp]
選項與參數：
-d ：後面可以接秒數，就是整個程序畫面更新的秒數。預設是 5 秒；
-p ：指定某些個 PID 來進行觀察監測而已。

在 top 執行過程當中輸入 ？ 可顯示 top 指令集
```

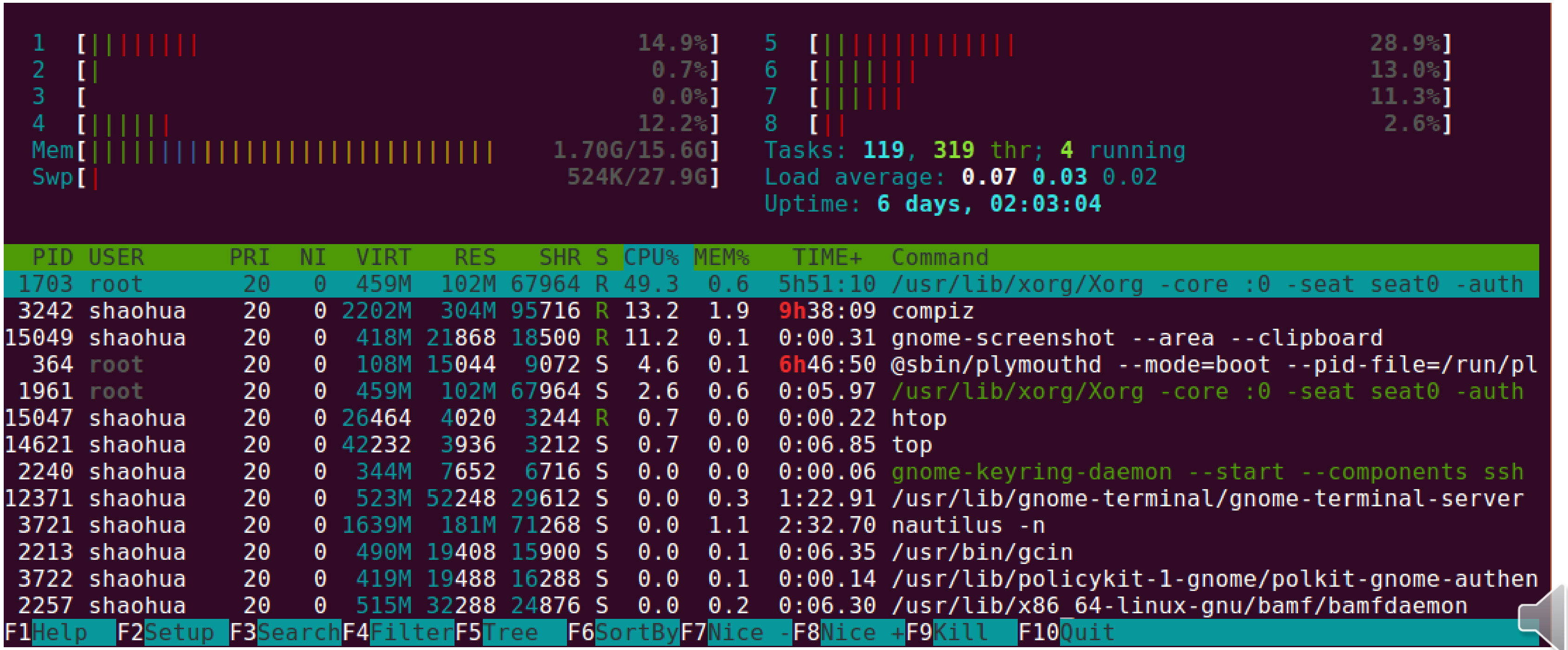
```
[root@study ~]# top
top - 00:53:59 up 6:07, 3 users, load average: 0.00, 0.01, 0.05
Tasks: 179 total, 2 running, 177 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2916388 total, 1839140 free, 353712 used, 723536 buff/cache
KiB Swap: 1048572 total, 1048572 free, 0 used. 2318680 avail Mem

  PID USER PR NI  VIRT  RES  SHR S %CPU %MEM   TIME+ COMMAND
18804 root 20 0 130028 1872 1276 R  0.5  0.1 0:00.02 top
    1 root 20 0  60636 7948 2656 S  0.0  0.3 0:01.70 systemd
    2 root 20 0      0    0    0 S  0.0  0.0 0:00.01 kthreadd
    3 root 20 0      0    0    0 S  0.0  0.0 0:00.00 ksoftirqd/0
```

Courtesy of top(1) - Linux man page

Process Observation Command - htop

- **htop** - interactive process viewer (more powerful tool than **top**)
  - It is similar to **top**, but allows you to scroll vertically and horizontally, so you can see all the processes running on the system, along with their full command lines
  - Tasks related to processes (e.g., killing, renicing) can be done without entering their PIDs
  - atop provides a more complete view of system activities for CPU, memory, swap, disks, network



1 [||||||||| 14.9%]
2 [| 0.7%]
3 [0.0%]
4 [|||||| 12.2%]
Mem[|||||||||||||||||||||||||||||| 1.70G/15.6G]
Swp[| 524K/27.9G]
5 [|||||||||||||| 28.9%]
6 [||||||| 13.0%]
7 [|||||| 11.3%]
8 [|| 2.6%]
Tasks: 119, 319 thr; 4 running
Load average: 0.07 0.03 0.02
Uptime: 6 days, 02:03:04
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1703 root 20 0 459M 102M 67964 R 49.3 0.6 5h51:10 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth
3242 shaohua 20 0 2202M 304M 95716 R 13.2 1.9 9h38:09 compiz
15049 shaohua 20 0 418M 21868 18500 R 11.2 0.1 0:00.31 gnome-screenshot --area --clipboard
364 root 20 0 108M 15044 9072 S 4.6 0.1 6h46:50 @sbin/plymouthd --mode=boot --pid-file=/run/pl
1961 root 20 0 459M 102M 67964 S 2.6 0.6 0:05.97 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth
15047 shaohua 20 0 26464 4020 3244 R 0.7 0.0 0:00.22 htop
14621 shaohua 20 0 42232 3936 3212 S 0.7 0.0 0:06.85 top
2240 shaohua 20 0 344M 7652 6716 S 0.0 0.0 0:00.06 gnome-keyring-daemon --start --components ssh
12371 shaohua 20 0 523M 52248 29612 S 0.0 0.3 1:22.91 /usr/lib/gnome-terminal/gnome-terminal-server
3721 shaohua 20 0 1639M 181M 71268 S 0.0 1.1 2:32.70 nautilus -n
2213 shaohua 20 0 490M 19408 15900 S 0.0 0.1 0:06.35 /usr/bin/gcin
3722 shaohua 20 0 419M 19488 16288 S 0.0 0.1 0:00.14 /usr/lib/policykit-1-gnome/polkit-gnome-authen
2257 shaohua 20 0 515M 32288 24876 S 0.0 0.2 0:06.30 /usr/lib/x86_64-linux-gnu/bamf/bamfdaemon
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit



Process status report

Process management

System resource monitor

PROCESS MANAGEMENT



Process Management with Signal

- Processes in Linux can ***control*** other processes through ***sending signals***
- Some commonly used signals are listed below

Singal Number	Signal Name	Meaning
1	**SIGHUP**	Hang up detected on controlling terminal or death of controlling process
2	**SIGINT**	Interrupt from keyboard
9	**SIGKILL**	Kill signal
15	**SIGTERM**	Termination signal

- The **SIGTERM** is the default signal sent by **kill** and **killall** commands
  - A well-designed application will implement the **SIGTERM** handler that is responsible for cleaning up the temporary files and releasing used resources to exit gracefully
- Unless you have **an unresponsive process**, you do not need to use **SIGKILL**

Courtesy of 第十六章、程序管理與 SELinux 初探 16.3.2 程序的管理, 鳥哥的 Linux 私房菜; signal(7) - Linux man page



Signal Sending Commands - kill & killall

- **kill** - terminate a process with its PID
- **killall** - terminate a process with its name

```
[root@study ~]# kill -signal PID
[root@study ~]# killall -signal [-i] [command name]
選項與參數：
-i ：interactive 的意思，互動式的，若需要刪除時，會出現提示字元給使用者；

範例一：強制終止 PID 8888 的程序
[root@study ~]# kill -9 8888

範例二：強制終止所有以 httpd 啟動的程序 (其實並沒有此程序在系統內)
[root@study ~]# killall -9 httpd

範例三：依次詢問每個 bash 程式是否需要被終止運作！
[root@study ~]# killall -i -9 bash
Signal bash(13888) ? (y/N) n <==這個不殺！
Signal bash(13928) ? (y/N) n <==這個不殺！
Signal bash(13970) ? (y/N) n <==這個不殺！
Signal bash(14836) ? (y/N) y <==這個殺掉！
# 若沒有 -i 的參數，所有的 bash 都會被殺掉，包括 root 自己的 bash。
```



Process Priority

- Each process is assigned a process priority
  - to determine how much CPU or processor time is allocated to it for execution
- Two kinds of priority in Linux: **Real-time priority** and **nice** values
  - **Real-time priority** (PR) goes from 1 to 99, w/ 100 to 139 dedicated to user-space
    - The process priority is adjusted by Linux dynamically; 1 is with the highest priority
  - The **nice** value (NI) has a range between -20 (highest priority) to +19 (lowest priority)
    - Only **nice** value can be adjusted by users, ranging from 0 ~ 19
  - **PRI**(new) = **PRI**(old) + **nice**

Priority and **nice** values of a process can be found in **top**

```
[root@study ~]# top
top - 00:53:59 up 6:07, 3 users, load average: 0.00, 0.01, 0.05
Tasks: 179 total, 2 running, 177 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2916388 total, 1839140 free, 353712 used, 723536 buff/cache
KiB Swap: 1048572 total, 1048572 free, 0 used. 2318680 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
18804 root      20   0  130028   1872   1276 R  0.5  0.1   0:00.02 top
    1 root      20   0   60636   7948   2656 S  0.0  0.3   0:01.70 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.01 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.00 ksoftirqd/0
```



Priority Adjustment Commands – nice & renice

- **nice** - run a program with modified scheduling priority

```
[root@study ~]# nice [-n 數字] command
選項與參數：
-n ：後面接一個數值，讓原本的 nice 加上這個新的數值之意。修改後的最終數值的範圍則為 -20 ~ 19。
範例一：用 root 讓原本的 nice 再減少 5 (-5)，用於執行 vim， 並觀察該程序！
[root@study ~]# nice -n -5 vim &
[1] 19865
[root@study ~]# ps -l
F S UID   PID  PPID C PRI NI ADDR SZ WCHAN  TTY        TIME CMD
4 S   0 14836 14835 0  90 10 - 29068 wait   pts/0 00:00:00 bash
4 T   0 19865 14836 0  85 -5 - 37757 signal pts/0 00:00:00 vim
0 R   0 19866 14836 0  90 10 - 30319 -      pts/0 00:00:00 ps
```

- **renice** - alter priority of running processes

```
[root@study ~]# renice [number] PID

範例一：找出自己的 bash PID， 並將該 PID 的 nice 調整到 -5
[root@study ~]# ps -l
F S UID   PID  PPID    C PRI NI ADDR SZ WCHAN TTY        TIME CMD
4 S   0  14836  14835  0  90 10 - 29068 wait  pts/0 00:00:00 bash

[root@study ~]# renice -5 14836
14836 (process ID) old priority 80, new priority -5

[root@study ~]# ps -l
F S UID   PID  PPID   C  PRI NI ADDR SZ WCHAN TTY        TIME CMD
4 S   0  14836  14835 0   85 -5  - 29068 wait  pts/0 00:00:00 bash
```

Courtesy of 第十六章、程序管理與 SELinux 初探 16.3.3 關於程序的執行順序, 鳥哥的 Linux 私房菜, kill(1) - Linux man page



Process status report

Process management

System resource monitor

PROCESS MANAGEMENT



System Resource Monitoring

- A computer has finite resources
  - such as CPU time, 4GB RAM, 1TB HDD storage, and network bandwidth
- Similarly, the number of processes is not limited
  - which would cause resource contention and poor performance
  - In fact, there is a number of max running processes in the system (4,194,303)
- Fully understand the use of resources can help us maximize performance



System Status – uname

- **uname** - print system information

```
[root@study ~]# uname [-asrmpi]
選項與參數：
-a ：所有系統相關的資訊，包括底下的資料都會被列出來；
-s ：系統核心名稱
範例一：輸出系統的基本資訊
[root@study ~]# uname –a
Linux study.centos.vbird 3.10.0-229.el7.x86_64 #1 SMP Fri Mar 6 11:36:42 UTC 2015
x86_64 x86_64 x86_64 GNU/Linux
```

- **lscpu** – display information about the CPU architecture

```
[root@study ~]# lscpu

範例一：輸出 CPU 的基本資訊
[root@study ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
....(中間省略)....
NUMA node0 CPU(s):     0-7
Flags:                 fpu vme de pse tsc msr pae mce cx8
CPU max MHz:           …
CPU min MHz:           …
apic ....(底下省略)....
```



Memory and Storage Status – free & df

- **free** - display amount of free and used memory in the system

```
[root@study ~]# free [-b|-k|-m|-g|-h] [-t] [-s N -c N]
選項與參數：
-b ：預設是 Kbytes， 可以使用 b(bytes), m(Mbytes) k(Kbytes), 及 g(Gbytes) 來顯示單位
-t ：在輸出的最終結果， 顯示實體記憶體與 swap 的總量。
-s ：每幾秒輸出一次
範例一：顯示目前系統的記憶體容量
[root@study ~]# free –m
     total used free shared buff/cache available
Mem:  2848  346 1794      8        706      2263
Swap: 1023    0 1023
```

- **df** - report file system disk space usage

```
[root@study ~]# df [-h]
選項與參數：
-h ：所有系統相關的資訊， 包括底下的資料都會被列出來；
[root@study ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.8G     0  7.8G   0% /dev
tmpfs           1.6G   18M  1.6G   2% /run
/dev/sdb1       321G  148G  158G  49% /          <==sdx 就是電腦硬碟
分割出的磁區
/dev/sdc2        96M   29M   68M  31% /boot/efi
```



Network Status – netstat

- **netstat** - print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

```
[root@study ~]# netstat -[atunlp]
選項與參數：
-a ：將目前系統上所有的連線、監聽、Socket 資料都列出來
-t ：列出 tcp 網路封包的資料
-u ：列出 udp 網路封包的資料
-n ：不以程序的服務名稱，以埠號 (port number) 來顯示；
-l ：列出目前正在網路監聽 (listen) 的服務；
-p ：列出該網路服務的程序 PID

範例一：找出目前系統上已在監聽的網路連線及其 PID
[root@study ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State  PID/Program name
tcp        0      0 0.0.0.0:22    0.0.0.0:*       LISTEN 1326/sshd
tcp        0      0 127.0.0.1:25  0.0.0.0:*       LISTEN 2349/master
tcp6       0      0 :::22         :::*            LISTEN 1326/sshd
tcp6       0      0 ::1:25        :::*            LISTEN 2349/master
Udp        0      0 0.0.0.0:123   0.0.0.0:*               751/chronyd
udp        0      0 127.0.0.1:323 0.0.0.0:*               751/chronyd
udp        0      0 0.0.0.0:57808 0.0.0.0:*               743/avahi-daemon: r
udp        0      0 0.0.0.0:5353  0.0.0.0:*               743/avahi-daemon: r
udp6       0      0 :::123        :::*                    751/chronyd
Udp6       0      0 ::1:323       :::*                    751/chronyd
# 除了可以列出監聽網路的介面與狀態之外，最後一個欄位還能夠顯示此服務的
# PID 號碼以及程序的指令名稱，例如上頭的 1326 就是該 PID

範例二：將上述的 0.0.0.0:57808 那個網路服務關閉的話？
[root@study ~]# kill -9 743 [root@study ~]# killall -9 avahi-daemon
```

```
範例三：列出目前系統已經建立的網路連線與 unix socket 狀態
[root@study ~]# netstat

Active Internet connections (w/o servers) <==與網路較相關的部分
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 172.16.15.100:ssh 172.16.220.234:48300 ESTABLISHED

Active UNIX domain sockets (w/o servers) <==與本機的程序相關(非網路)
Proto RefCnt Flags Type State I-Node Path
unix 2 [ ] DGRAM 1902 @/org/freedesktop/systemd1/notify
unix 2 [ ] DGRAM 1944 /run/systemd/shutdownd
....(中間省略)....
unix 3 [ ] STREAM CONNECTED 25425 @/tmp/.X11-unix/X0
unix 3 [ ] STREAM CONNECTED 28893
unix 3 [ ] STREAM CONNECTED 21262
```

Courtesy of 第十六章、程序管理與 SELinux 初探 16.3.4 系統資源的觀察, 鳥哥的 Linux 私房菜;



THANK YOU!!!