HỆ ĐIỀU HÀNH
Chủ đề 1: Tổng quan

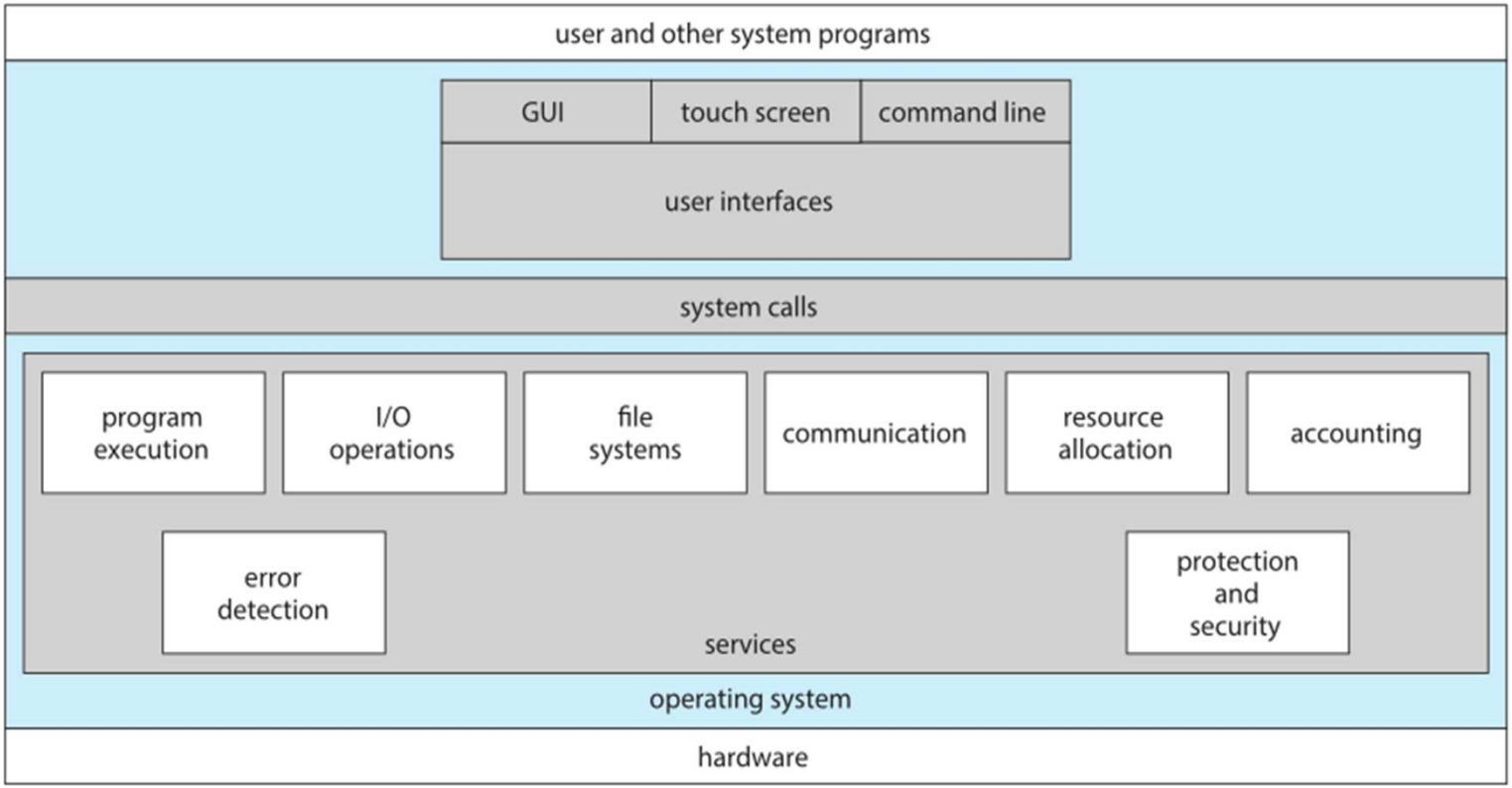
Trường Đại học Nha Trang
Khoa Công nghệ thông tin
Bộ môn Hệ thống thông tin
Giáo viên: Ts.Nguyễn Khắc Cường

CHƯƠNG 2

CẤU TRÚC HỆ ĐIỀU HÀNH

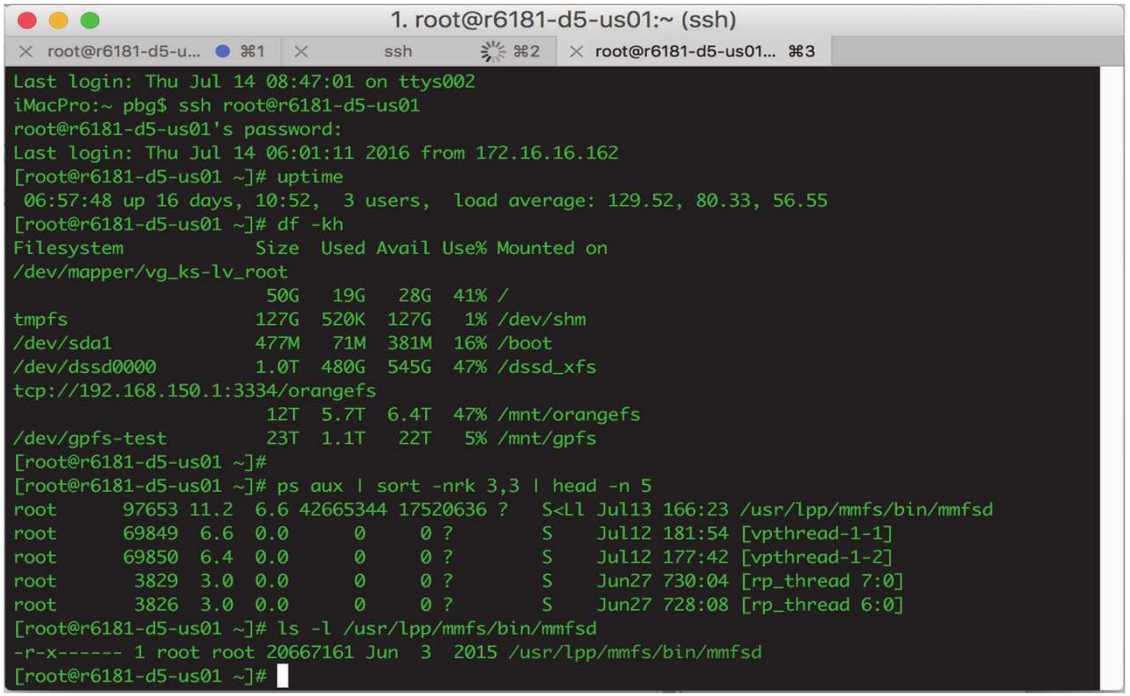
Các service của HĐH

- Các service là khác nhau trong các loại HĐH, nhưng các HĐH đều cung cấp các nhóm service cơ bản như sau:



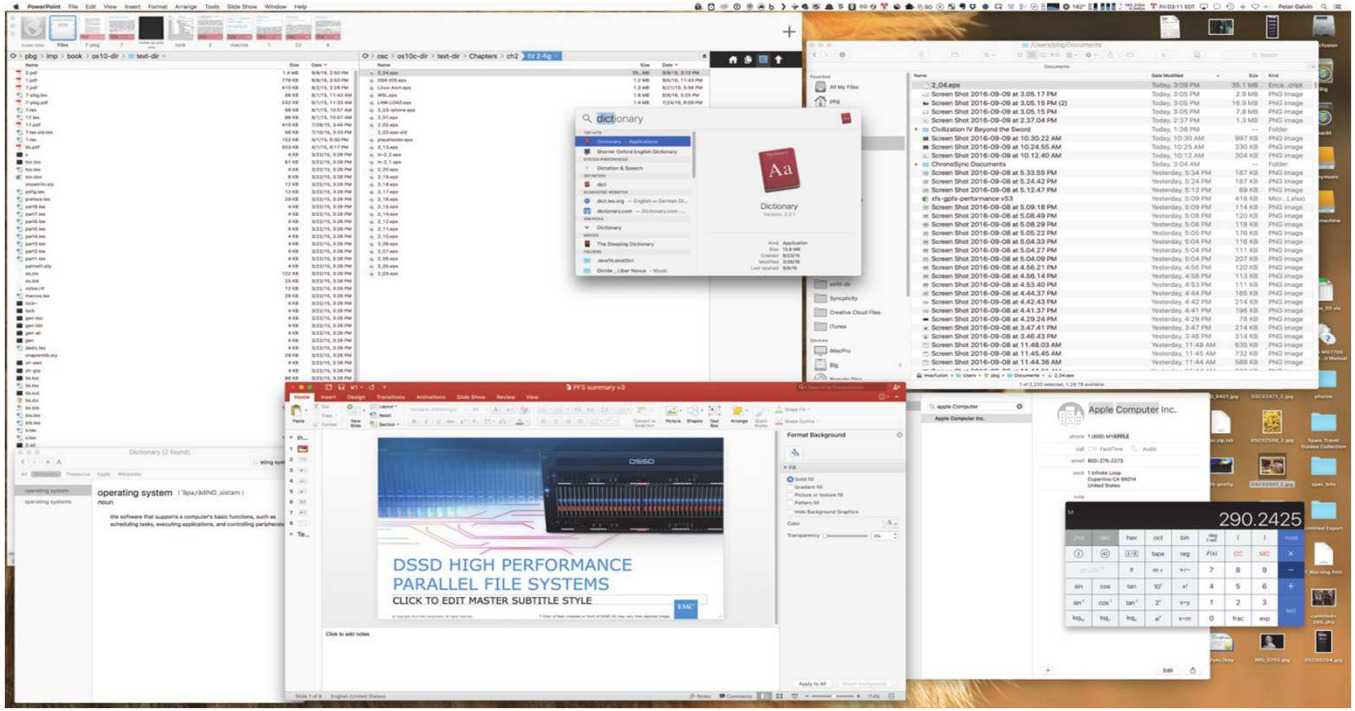
Các kiểu giao diện của HĐH

- Command interpreter
  - Các HĐH như Linux, UNIX, Windows đều có cung cấp giao diện này
  - Giao diện này còn có tên là Shell
  - Các nhà phát triển Linux, UNIX đã phát triển nhiều loại shell như:
    - C shell, Bourne-Again shell, Korn shell, . . .



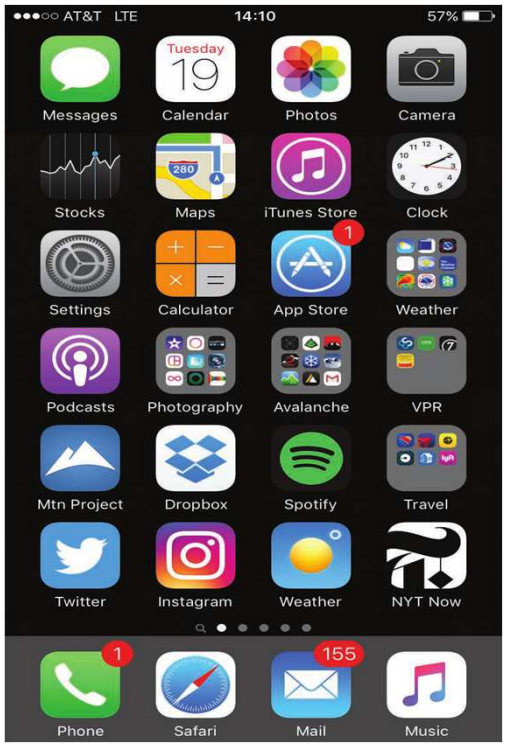
Các kiểu giao diện của HĐH

- Graphical User Interface (GUI)
  - Tuy Linux và UNIX được các user vận hành chủ yếu bằng giao diện cửa số lệnh, nhưng hiện nay các nhà phát triển của các HĐH này cũng xây dựng các GUI và chia sẻ ở dạng mã nguồn mở, ví dụ như: K Desktop Environment (KDE), GNOME desktop của dự án GNU, …



Các kiểu giao diện của HĐH

- Touch-Screen Interface
  - Là giao diện dùng màn hình cảm ứng
  - User ra lệnh cho OS bằng cách chạm ngón tay trực tiếp vào màn hình
  - Ví dụ, iPad và iPhone sử dụng giao diện màn hình cảm ứng tên là Springboard



Lựa chọn giao diện

- Chọn giao diện nào là do sở thích của user và đặc điểm công việc
- VD:
  - Với các system administrator
    - Thường phải quản lý một số lượng lớn các máy tính
    - Thường ra lệnh ở mức hệ thống (mức kernel)

    → Giao diện cửa sổ lệnh giúp thay đổi các tham số dễ dàng hơn, các lệnh thường xuyên thực hiện được lưu ở dạng shell script giúp thực hiện một số lượng lớn các lệnh chỉ bằng thao tác đơn giản là chạy cả file script
  - Với các user của OS Windows:
    - GUI là giao diện được ưu chuộng hơn
  - Với các thiết bị mobile
    - Touch-screen được lựa chọn vì tiện lợi cho user và nhà sản xuất

System calls

- Đây là một công cụ do HĐH cung cấp giúp user có thể thực hiện các service trong kernel của OS
- System call là công cụ hỗ trợ hiệu quả lập trình ứng dụng, lập trình hệ thống, lập trình điều khiển phần cứng, lập trình nhúng, …
- System call có thể được gọi trong các program viết bằng ngôn ngữ lập trình C, C++, Assembly, …
- Các system call có thể được gọi bằng các cách sau:
  - Gọi trực tiếp một function cụ thể của system call
  - Gọi thông qua các API (Application Programming Interface)

System calls

- Sự khác nhau giữa hai cách gọi system call
  - Gọi trực tiếp một function cụ thể của system call
    - Giả sử cần viết một program để thực hiện thao tác sao chép file, program đó cần gọi các system call sau:

Example System-Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally

System calls

- Sự khác nhau giữa hai cách gọi system call
  - Gọi thông qua các API
    - Một API được thiết kế để thực hiện một tập hợp các system call thay vì chỉ một system call để thực hiện một tác vụ nào đó → nhằm giảm bớt sự phức tạp của program, giúp phát triển các ứng dụng được dễ dàng hơn. Cụ thể là:
      - Khả năng portability
      - User chỉ khai báo các tham số
    - Các API phổ biến nhất hiện nay gồm:
      - Windows API trong HĐH Windows
      - POSIX API trong các HĐH UNIX, Linux, macOS
      - Java API trong Java virtual machine
    - Cú pháp của các API được các HĐH cung cấp cụ thể trong các thư viện. VD: Thông tin các API của OS UNIX và Linux có thể xem trong thư viện tên "libc", viết bằng ngôn ngữ lập trình C

Sự phụ thuộc của ứng dụng vào OS

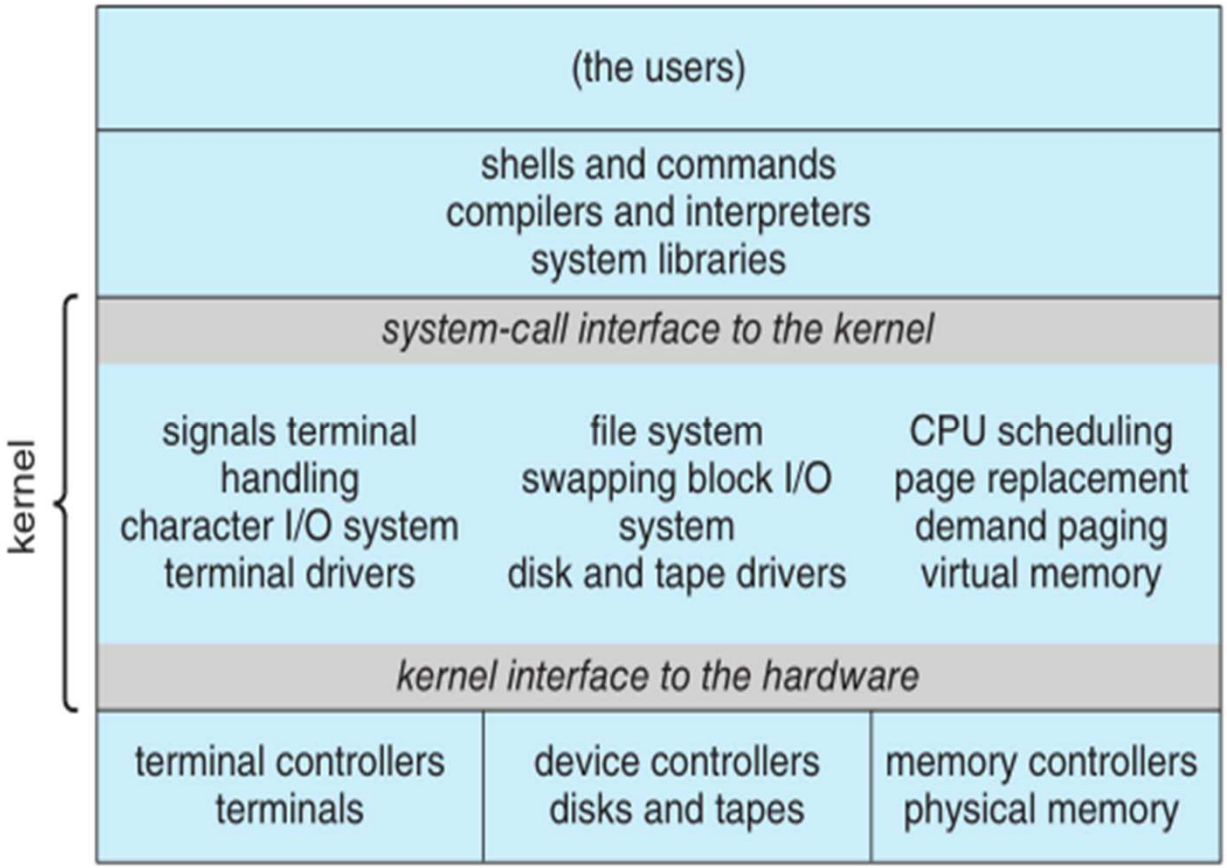
- Thông thường, một program được tạo ra trong HĐH nào thì chỉ complie và load được trong HĐH đó
- Hiện nay, nếu muốn tạo ra một program có thể compile và load được ở nhiều HĐH khác nhau thì có thể sử dụng các cách sau:
  - Sử dụng ngôn ngữ lập trình Python và Ruby
  - Viết code trong một máy ảo được hỗ trợ bởi nhiều HĐH khác nhau. VD: JAVA virtual machine được hỗ trợ bởi nhiều HĐH khác nhau
  - Sử dụng API được hỗ trợ bởi nhiều HĐH. VD:
    - Dùng POSIX API để dùng được trong các OS UNIX, Linux, mac OS

Thiết kế và phát triển HĐH

- Trong thực tế:
  - Không có giải pháp để tạo ra HĐH hoàn hảo.
  - Chỉ có các giải pháp đủ tốt cho một hệ thống nào đó để đáp ứng nhu cầu cụ thể nào đó thôi. Khi yêu cầu phát sinh thì HĐH lại được nâng cấp thêm.
- Việc thiết kế và phát triển một HĐH phụ thuộc vào các yếu tố cơ bản sau:
  - Design goals
  - Mechanisms and Policies
  - Implementation

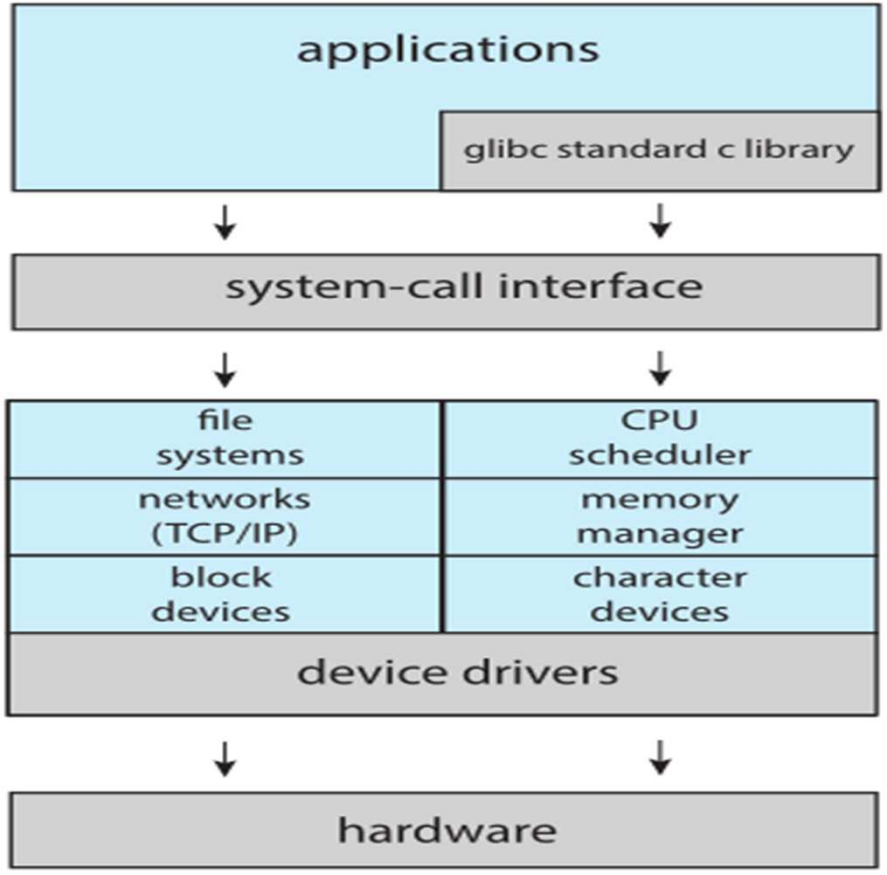
Cấu trúc của OS

- Cấu trúc phân lớp
  - Cấu trúc phân lớp của HĐH UNIX



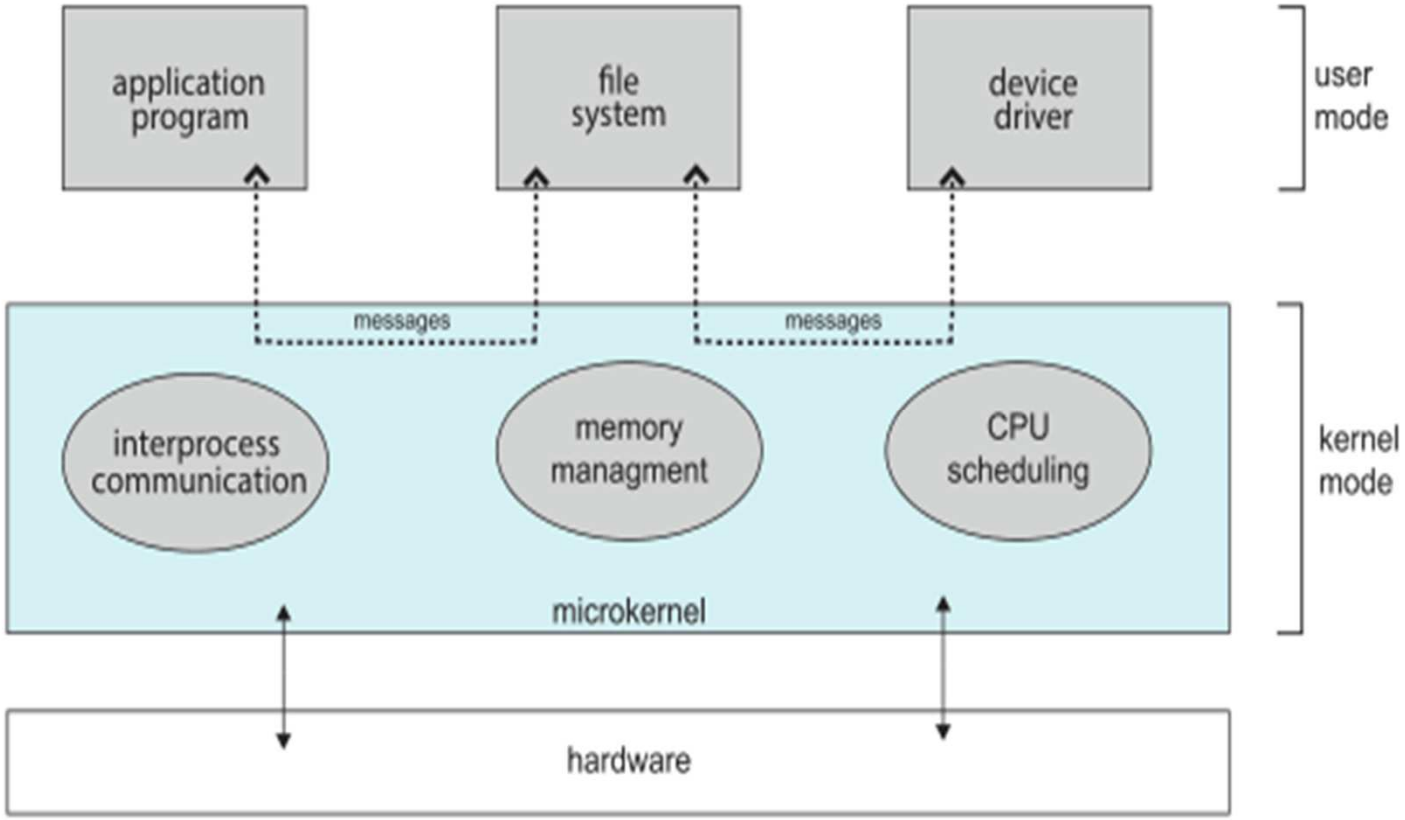
Cấu trúc của OS

- Cấu trúc phân lớp
  - Cấu trúc phân lớp của HĐH LINUX



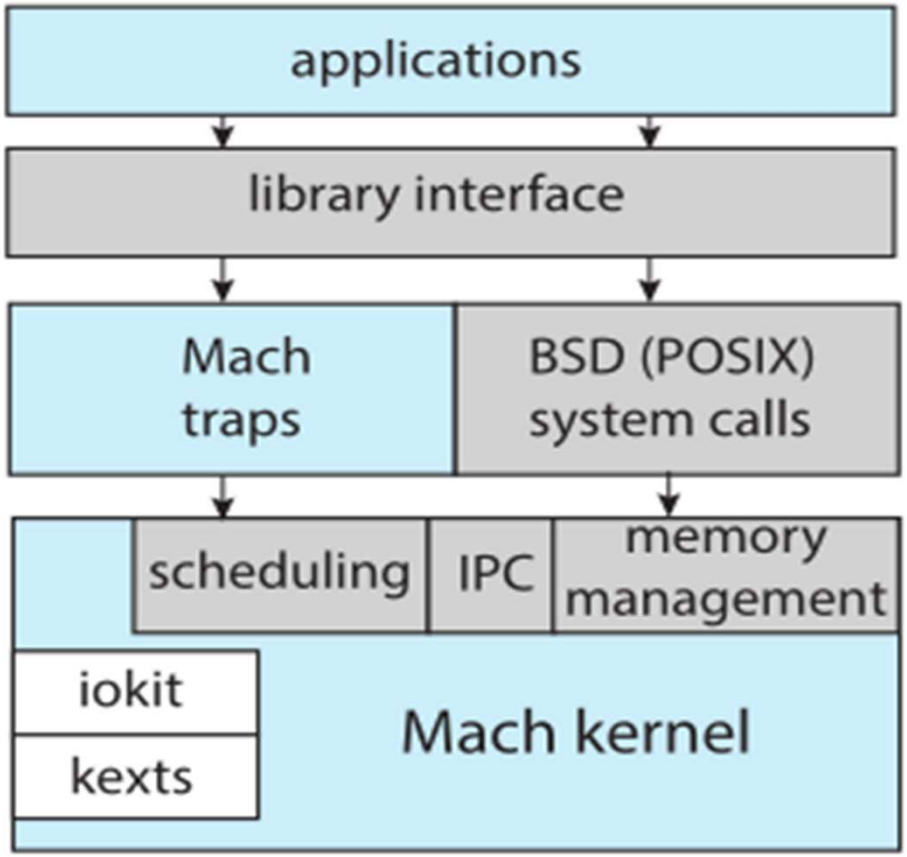
Cấu trúc của OS

- Cấu trúc phân lớp
  - Kernel của HĐH cũng có cấu trúc phân lớp



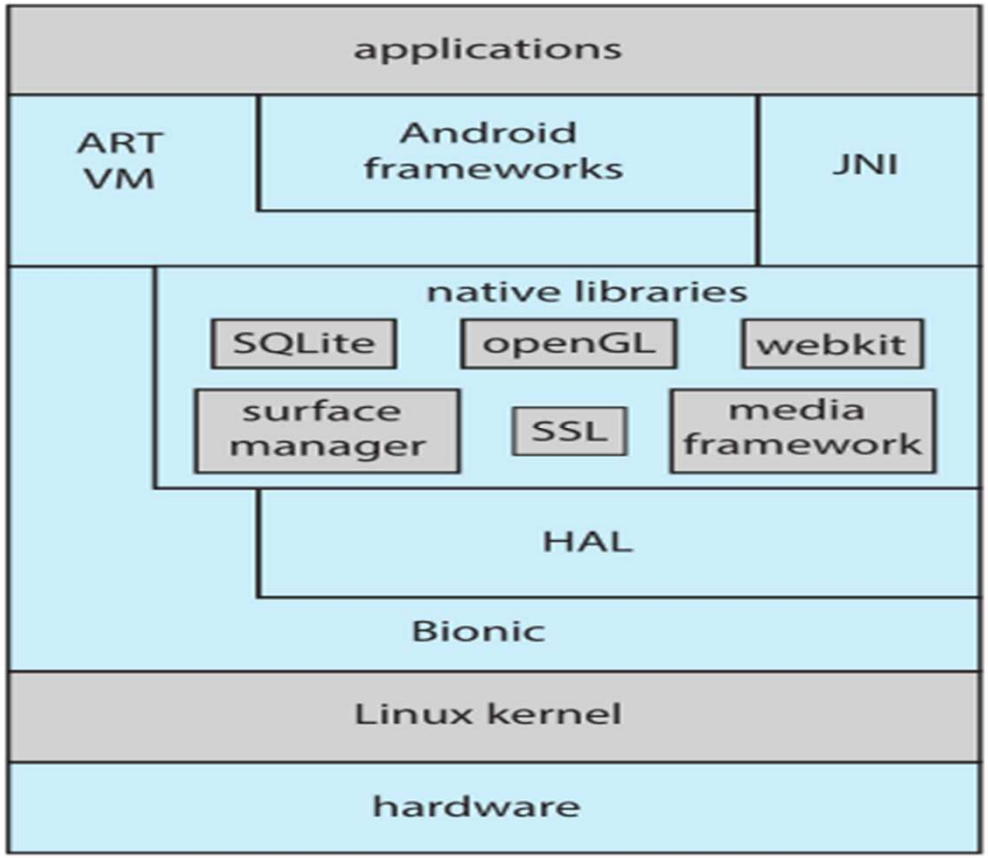
Cấu trúc của OS

- Cấu trúc phân lớp
  - Cấu trúc phân lớp của kernel Darwin (macOS của Apple)



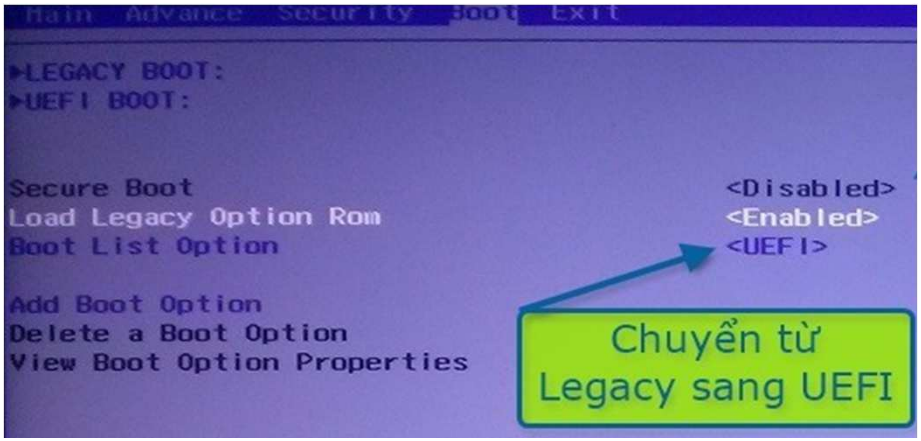
Cấu trúc của OS

- Cấu trúc phân lớp
  - Cấu trúc phân lớp của Android



Sơ lược các bước cơ bản trong qui trình khởi động của các loại HĐH

- Qui trình khởi động OS được gọi là booting
- Booting xảy ra khi máy tính mở điện hay khởi động lại
- Một số khái niệm liên quan đến quá trình booting:
  - Bootstrap (Boot loader)
  - BIOS
- Các bước cơ bản của quá trình booting
- UEFI (Unified Extensible Firmware Interface)



Debugging

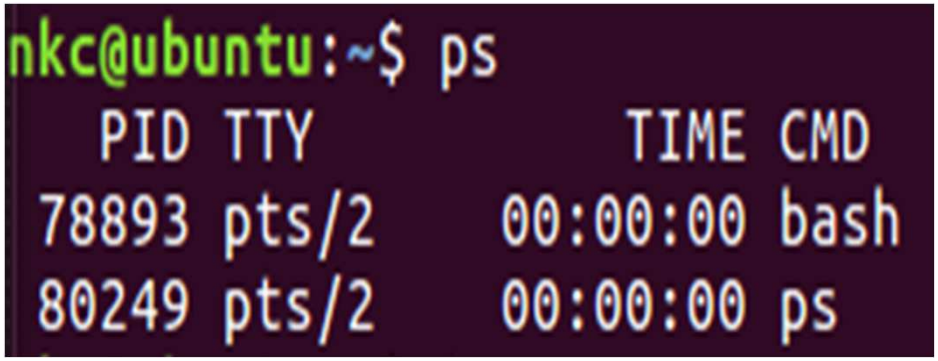
- Debugging là các hoạt động HĐH thực hiện để phát hiện và sửa lỗi phát sinh trong quá trình quản lý và vận hành
- Tác dụng của debugging là để loại bỏ các bottleneck nhằm nâng cao hiệu năng của hệ thống
- Debugging có thể thực hiện như sau:
  - Phân tích lỗi (failure analysis)
    - Khi một tiến trình nào đó ở user mode xảy ra lỗi
    - Khi một tiến trình nào đó ở kernel mode xảy ra lỗi
    - Nơi cung cấp thông tin cho quá trình phân tích lỗi

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Các phương pháp theo dõi hiệu năng được chia thành 2 nhóm chính:
      - Per-process
      - System-wide
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - Tác dụng
        - VD: một số công cụ counters trong OS Linux

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - VD: một số công cụ counters trong OS Linux
          - Counters theo dõi kiểu Per-process: ps



Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - VD: một số công cụ counters trong OS Linux
          - Counters theo dõi kiểu Per-process: top

```
top - 20:11:43 up 1 day, 14:47,  1 user,  load average: 0.06, 0.09, 0.05
Tasks: 243 total,   1 running, 171 sleeping,   0 stopped,   2 zombie
%Cpu(s):  0.8 us,  0.8 sy,  0.0 ni, 97.5 id,  0.8 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2017552 total,   614176 free,   555488 used,   847888 buff/cache
KiB Swap:   998396 total,   602108 free,   396288 used.  1207472 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3610 nkc       9 -11  584576   6564   5044 S   1.0  0.3  32:20.59 pulseaudio
  904 root      20   0  468612  42700  14552 S   0.7  2.1   3:19.32 Xorg
 1770 nkc       20   0  365512   4728   3512 S   0.3  0.2   0:01.97 ibus-daemon
 1869 nkc       20   0  208684   3092   2784 S   0.3  0.2   0:00.50 ibus-engine-sim
 3828 nkc       20   0 1279692  62840  36224 S   0.3  3.1  16:24.67 compiz
 4746 nkc       20   0  673136  25668  18204 S   0.3  1.3   0:11.92 gnome-terminal-
80250 nkc       20   0   41948   3884   3152 R   0.3  0.2   0:00.02 top
    1 root      20   0  185388   4748   3180 S   0.0  0.2   0:07.54 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.03 kthreadd
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S   0.0  0.0   0:01.87 ksoftirqd/0
    8 root      20   0       0      0      0 I   0.0  0.0   2:42.58 rcu_sched
    9 root      20   0       0      0      0 I   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.12 migration/0
```

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - VD: một số công cụ counters trong OS Linux
          - Counters theo dõi kiểu System-wide: vmstat

```
nkc@ubuntu:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 395520 574476 145312 703576    1    4    18    44    3  151 10  6 84  0  0
```

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - VD: một số công cụ counters trong OS Linux
          - Counters theo dõi kiểu System-wide: netstat

```
nkc@ubuntu:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    29856    /run/user/1000/systemd/notify
unix  2      [ ]         DGRAM                    19711    /run/systemd/cgroups-agent
unix  2      [ ]         DGRAM                    19722    /run/systemd/journal/syslog
unix  7      [ ]         DGRAM                    19729    /run/systemd/journal/socket
unix  16     [ ]         DGRAM                    19744    /run/systemd/journal/dev-log
unix  3      [ ]         DGRAM                    19710    /run/systemd/notify
unix  3      [ ]         STREAM     CONNECTED     36683    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     34240
unix  3      [ ]         STREAM     CONNECTED     23302    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     35604
unix  3      [ ]         STREAM     CONNECTED     24001    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     34621    @/tmp/dbus-6V83dsidRz
unix  3      [ ]         STREAM     CONNECTED     32710
```

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - VD: một số công cụ counters trong OS Linux
          - Counters theo dõi kiểu System-wide: iotstat

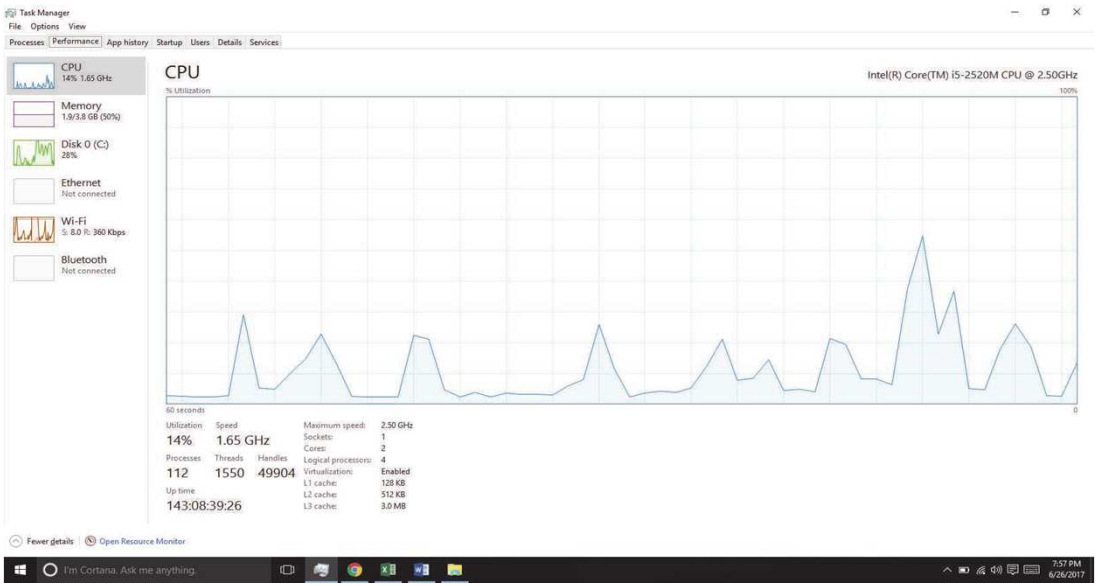
```
nkc@ubuntu:~$ iostat
Linux 4.15.0-112-generic (ubuntu)      08/02/2021      _x86_64_       (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          9.71    0.14    5.74    0.17    0.00   84.23

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
loop0             0.19         0.19         0.00      26691          0
loop1             0.00         0.00         0.00        572          0
loop2             0.00         0.00         0.00        386          0
loop3             0.00         0.00         0.00          1          0
sda               2.43        36.07        85.89    5048922   12023672
```

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Counters
        - VD: Windows cung cấp công cụ Windows Task Manager có chức năng tương tự



Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Tracing
        - Tác dụng
        - VD: một số công cụ tracing trong OS Linux
          - Tracing kiểu Per-process: strace

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Tracing
        - Tác dụng
        - VD: một số công cụ tracing trong OS Linux
          - Tracing kiểu Per-process: gdb

```
nkc@ubuntu:~$ gdb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)
```

Debugging

- Debugging có thể thực hiện như sau:
  - Theo dõi hiệu năng của hệ thống và hiệu chỉnh (performance monitoring and tuning)
    - Trong mỗi phương pháp, có thể sử dụng các công cụ sau:
      - Tracing
        - Tác dụng
        - VD: một số công cụ tracing trong OS Linux
          - Tracing kiểu System-wide: perf

```
nkc@ubuntu:~$ perf

usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]

The most commonly used perf commands are:
   annotate        Read perf.data (created by perf record) and display annotated code
   archive         Create archive with object files with build-ids found in perf.data file
   bench           General framework for benchmark suites
   buildid-cache   Manage build-id cache.
   buildid-list    List the buildids in a perf.data file
   c2c             Shared Data C2C/HITM Analyzer.
   config          Get and set variables in a configuration file.
   data            Data file related processing
   diff            Read perf.data files and display the differential profile
   evlist          List the event names in a perf.data file
   ftrace          simple wrapper for kernel's ftrace functionality
   inject          Filter to augment the events stream with additional information
   kallsyms        Searches running kernel for symbols
   kmem            Tool to trace/measure kernel memory properties
   kvm             Tool to trace/measure kvm guest os
   list            List all symbolic event types
   lock            Analyze lock events
   mem             Profile memory accesses
   record          Run a command and record its profile into perf.data
   report          Read perf.data (created by perf record) and display the profile
   sched           Tool to trace/measure scheduler properties (latencies)
   script          Read perf.data (created by perf record) and display trace output
   stat            Run a command and gather performance counter statistics
   test            Runs sanity tests.
   timechart       Tool to visualize total system behavior during a workload
   top             System profiling tool.
   probe           Define new dynamic tracepoints
```

Debugging

- Việc phát triển các công cụ theo dõi hiệu năng vận hành của OS và đề xuất các phương pháp mới để hiệu chỉnh hiệu năng sao cho OS hoạt động có hiệu quả cao hơn vẫn đang là một lĩnh vực nghiên cứu rất rộng lớn, sôi động và có nhiều người nghiên cứu hiện nay

Debugging

- Dùng công cụ để debug các tương tác giữa user mode và kernel
  - Debug các tương tác giữa user mode và kernel gần như là không thể thực hiện được vì các tương tác này:
    - có số lượng rất lớn
    - rất phức tạp
    - không thể chỉnh sửa tùy tiện vì dễ gây lỗi hay ảnh hưởng đến hiệu năng của hệ thống đang hoạt động
  - Tuy nhiên, hiệu năng của hệ thống cũng được quyết định rất nhiều bởi các tương tác này
  - Do đó, rất cần một công cụ có thể giúp debug được các tương tác giữa user mode và kernel một cách linh động, an toàn, không làm giảm hiệu năng của hệ thống trong khi đang thực hiện debug

Debugging

- Dùng công cụ để debug các tương tác giữa user mode và kernel
  - Trong OS Linux, BCC là một công cụ đáp ứng được các yêu cầu trên
    - BCC = BPF Compiler Collection = là front-end interface của công cụ eBPF (extended Berkeley Packet Filter)
    - eBPF được viết bằng ngôn ngữ C nên việc phát triển thêm các công cụ cho eBPF khá phức tạp
    - BCC là một giao diện front-end eBPF và được viết bằng Python để tạo điều kiện dễ dàng hơn trong việc phát triển thêm các công cụ cho BCC
    - Sau khi cài đặt BCC vào Linux, có thể dùng các công cụ của BCC để debug các tương tác trong kernel của Linux

Debugging

- Dùng công cụ để debug các tương tác giữa user mode và kernel
  - Trong OS Linux, BCC là một công cụ đáp ứng được các yêu cầu trên
  - Các công cụ của BCC

```
nkc@ubuntu:/usr/share/bcc/tools$ ls
argdist       dbstat       gethostlatency  nfsslower    pythonflow  sslsniff     tcptop
bashreadline  dcsnoop      hardirqs        nodegc       pythongc    stackcount   tcptracer
biolatency    dcstat       inject          nodestat     pythonstat  statsnoop    tplist
biosnoop      deadlock     javacalls       offcputime   reset-trace syncsnoop    trace
biotop        deadlock.c   javaflow        offwaketime  rubycalls   syscount     ttysnoop
bitesize      doc          javagc          old          rubyflow    tclcalls     vfscount
bpflist       drsnoop      javaobjnew      oomkill      rubygc      tclflow      vfsstat
btrfsdist     execsnoop    javastat        opensnoop    rubyobjnew  tclobjnew    wakeuptime
btrfsslower   exitsnoop    javathreads     perlcalls    rubystat    tclstat      xfsdist
cachestat     ext4dist     killsnoop       perlflow     runqlat     tcpaccept    xfsslower
cachetop      ext4slower   lib             perlstat     runqlen     tcpconnect   zfsdist
capable       filelife     llcstat         phpcalls     runqslower  tcpconnlat   zfsslower
cobjnew       fileslower   mdflush         phpflow      shmsnoop    tcpdrop
cpudist       filetop      memleak         phpstat      slabratetop tcplife
cpuunclaimed  funccount    mountsnoop      pidpersec    sofdsnoop   tcpretrans
criticalstat  funclatency  mysqld_qslower  profile      softirqs    tcpstates
dbslower      funcslower   nfsdist         pythoncalls  solisten    tcpsubnet
```

Debugging

- Dùng công cụ để debug các tương tác giữa user mode và kernel
  - VD một công cụ của BCC là: biosnoop

```
^Cnkc@ubuntu:/usr/share/bcc/tools$ sudo ./biosnoop
TIME(s)     COMM           PID    DISK    T SECTOR     BYTES   LAT(ms)
0.000000    ?              0              R 0          8          0.26
0.005213    biosnoop       86949  sda     R 13598952   4096       0.28
0.063837    ?              0              R 0          8          0.14
2.015695    ?              0              R 0          8          0.07
2.079832    ?              0              R 0          8          0.19
4.032089    ?              0              R 0          8          0.25
4.095724    ?              0              R 0          8          0.07
5.696527    jbd2/sda1-8    295    sda     W 17264624   8192       0.55
5.697194    jbd2/sda1-8    295    sda     W 17264640   4096       0.50
6.047805    ?              0              R 0          8          0.11
6.111779    ?              0              R 0          8          0.10
8.064102    ?              0              R 0          8          0.14
8.127779    ?              0              R 0          8          0.11
10.079772   ?              0              R 0          8          0.08
10.143802   ?              0              R 0          8          0.11
12.095731   ?              0              R 0          8          0.08
12.159818   ?              0              R 0          8          0.09
14.111737   ?              0              R 0          8          0.08
14.175773   ?              0              R 0          8          0.09
15.032147   dhclient       86585  sda     W 38034024   4096      18.63
15.032903   jbd2/sda1-8    295    sda     W 17264648   16384      0.34
15.033147   jbd2/sda1-8    295    sda     W 17264680   4096       0.19
16.127755   ?              0              R 0          8          0.10
16.191798   ?              0              R 0          8          0.11
18.143740   ?              0              R 0          8          0.08
18.207853   ?              0              R 0          8          0.13
20.159826   ?              0              R 0          8          0.11
20.223735   ?              0              R 0          8          0.07
20.544534   jbd2/sda1-8    295    sda     W 17264688   8192       0.61
20.545258   jbd2/sda1-8    295    sda     W 17264704   4096       0.33
22.175765   ?              0              R 0          8          0.08
22.239846   ?              0              R 0          8          0.12
23.616240   kworker/u256:0 86862  sda     W 34036448   4096       0.44
23.616354   kworker/u256:0 86862  sda     W 38034120   4096       0.48
23.616696   kworker/u256:0 86862  sda     W 38034552   4096       0.77
```

Debugging

- Dùng công cụ để debug các tương tác giữa user mode và kernel
  - Tóm tắt các công cụ BCC