COMSW 4118-2-1 9/7/2017

- What is an Operating System (cont. from last time)
  How would you define it? What gets included in an OS?
  Is a web browser part of the OS (ex. MSIE)?
  - ▶ You can delete the browser from the OS and it will still function!
  - ▶ But MS argued there are some key functions in it (they should have integrated it more tightly).

---

① ● One aspect is the structure of the software itself:

APPLICATIONS (safari) ← But, what defines this line

OS ← manages hardware

HARDWARE (CPU, RAM)

- You can think of the OS as two flavors:
  - ▶ OS = operating system environment
    = OS distribution (ex. Ubuntu w/ all things included) ← utilities, shell, browser
  - ▶ OS = kernel only (the core that manages the hardware, ex. Linux)

The kernel controls the hardware, thus has "extra power"

- What's in the kernel? It depends, different philosophies
  - ▶ Linux = monolithic kernel, includes every thing that manages hardware
    Problem: the bigger the software, the more bugs.
  - ▶ Microkernel, kernel is as small as possible, as much functionality moved out.
    Problem: outside functions may still need to go to kernel anyway = overhead

Both systems still exist: Linux is monolithic but modularized, MacOS & WinNT are (kind of) microkernel w/ minimized overhead (by expanding kernel). ex. including graphics support.

- How do we empower the kernel?
  It can do some things that you can't outside the kernel:
  - ▶ Privileged instructions - kernel can run, apps can't
    The hardware will burn apps that try to run privileged instructions
    The hardware defines kernel mode and user mode
    ↑ can run privileged ↑ can't run privileged

---

② ● Another aspect to think of an OS is as providing services to applications.
Applications are written on top of the OS to factor out common functions in the OS itself, to make the apps relatively portable.
Ex: I/O (file abstraction), ex: read/write (syscalls) or fread/fwrite (library),
Linux kernel folder arch has sub-folders w/ CPU-specific implementations

COMSW4118-2-2 9/7/2017

- Operating System Calls - i.e. system calls - the API OS provides to apps. (ex. read / write)

  Process - abstraction of a program in execution that allows the OS to run programs

- Process: you create a process w/ a system call w/ params, ex:

  create_process(program, resources, precedence, memory, files, ...)

  Linux/Unix uses simply (copying the current process):

  fork() $A \xrightarrow[\text{fork}]{} A'$ and $A \approx A'$

  - After fork() you have two processes: parent (fork returns child pid), and child (fork returns 0); getpid() to get own pid.