

OS Organization

Operating Systems

SAELEE:CS450-S12,#2

“

operating system

noun

the software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.

New Oxford American Dictionary

internal responsibilities
vs.
external API

managing **resources**

vs.

servicing **requests**

requests = syscalls

Resources

1.CPU

2.Memory

3.I/O Devices

OS and the hardware/ISA

1. CPU

scheduling & accounting

interrupt mechanism

2. Memory

virtual memory / swapper

memory **m**anagement **u**nit

3. I/O

uniform API → device drivers

physical device/bus,
memory mapped I/O

a unifying abstraction:
the **Process**

process management:
multitasking

context switches

Strategies

1. ~~Batch execution~~
2. Cooperative multitasking
3. Preemptive multitasking
4. Real-time systems

2. voluntary yielding

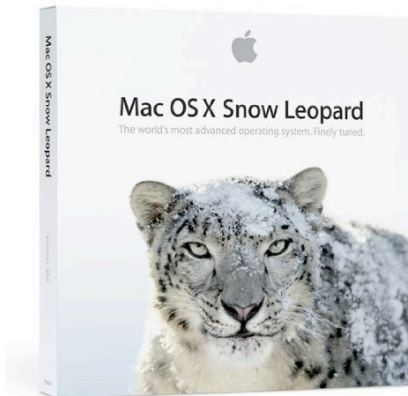
3. OS *preempts*

“cop” process



4. real time constraints

OS architecture



what's in an operating system?

e.g., “GNU/Linux”

“

A Unix-like operating system is a software collection of applications, libraries, and developer tools, plus a program to allocate resources and talk to the hardware, known as a kernel.

... GNU is typically used today with a kernel called Linux. This combination is the GNU/Linux operating system. GNU/Linux is used by millions, though many call it "Linux" **by mistake**.

GNU.org homepage

“

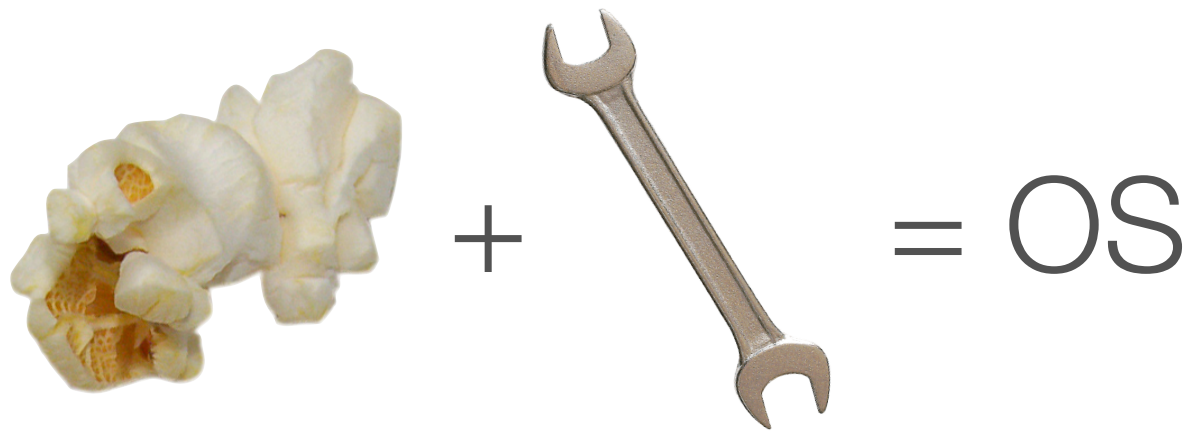
Why do you call it GNU/Linux and not Linux?

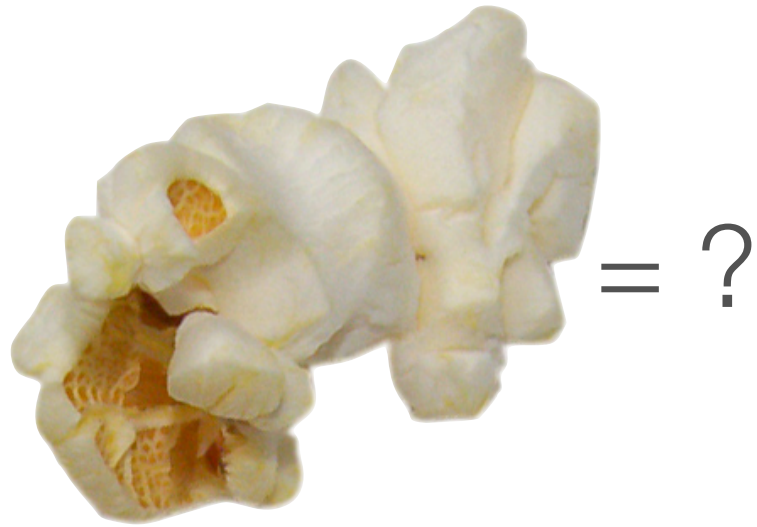
Most operating system distributions based on Linux as kernel are basically modified versions of the **GNU operating system**. We began developing GNU in 1984, years before Linus Torvalds started to write his kernel. ... We developed most of the central components, forming the largest single contribution to the whole system.

Should we always say “GNU/Linux” instead of “Linux”?

Not always—only when you're talking about the whole system. When you're referring specifically to the kernel, you should call it “Linux”, the name its developer chose.

GNU.org FAQ





OS **Kernel**

goal 1:
protect system from faults

goal 2:
protect system from
malicious behavior

goal 3:
optimize **performance**

goal 4:
allow separation of
mechanism and **policy**

hardware support:
supervisor mode flag
restricts access to hardware
& privileged instructions

protected “user” mode

vs.

supervisor “kernel” mode

transition: trap/interrupt

kernel decision:
what executes in
privileged mode?

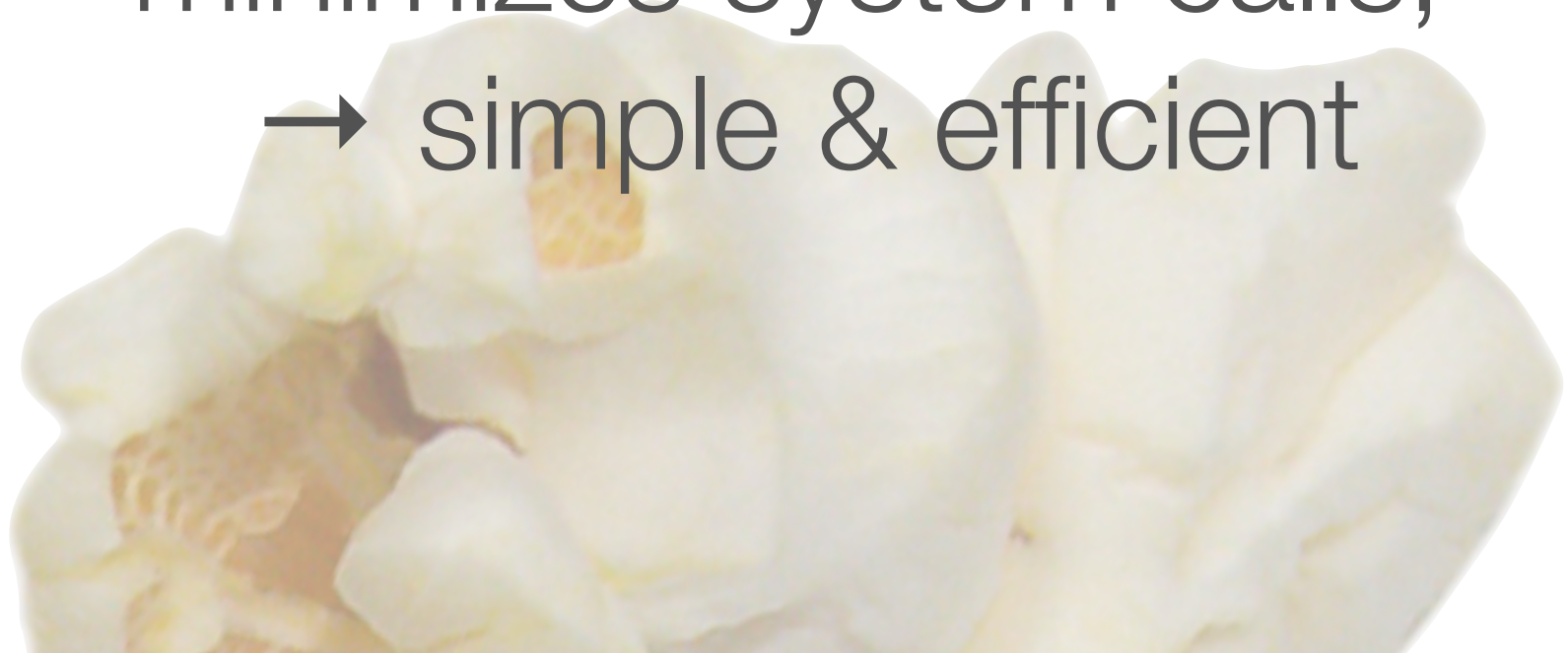


organization

monolithic
everything privileged



single address space,
minimizes system calls,
→ simple & efficient



(possible) drawbacks:
reduced robustness,
large memory footprint



micro 🍿

minimal implementation

OS modules → user services



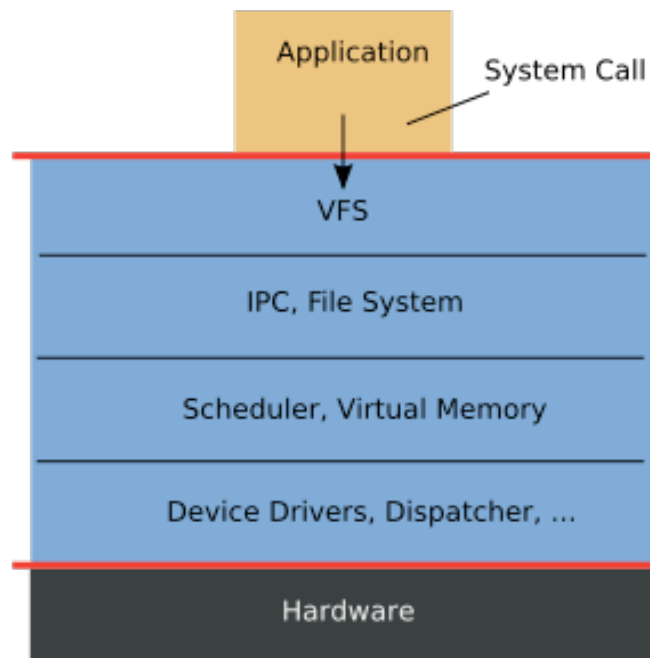


separates policy & mechanism,
improved robustness

lots of system calls,

 = “glorified message queue”

Monolithic Kernel based Operating System



Microkernel based Operating System

user mode

kernel mode

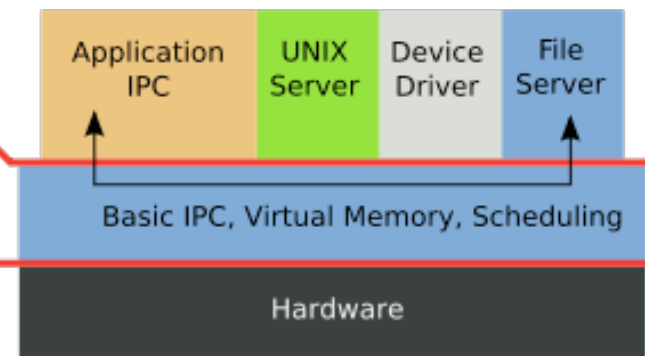


figure courtesy Wikimedia Commons

reality: **hybrid** kernel
architectures are common

“

...suffice it to say that among the people who actually design operating systems, the debate is essentially over.

Microkernels have won

Andrew Tanenbaum

“

The whole "microkernels are simpler" argument is just **bull**, and it is clearly shown to be bull by the fact that whenever you compare the speed of development of a microkernel and a traditional kernel, the **traditional kernel wins**.

By a huge amount, too.

Linus Torvalds

your opinion?
→ assignment 1