SYSTEMS ENGINEERING · WEEK 1

What is Selfie?

The talk, then the spine of the semester in an hour — weighted to the third command.

WHAT THIS CLASS IS FOR

Isolation is the semantic problem of systems. Virtualization buys performance by introducing self-reference. What a kernel cannot decide, it bounds.

Three sentences, and the class is their derivation. Fourteen weeks of building hypster, the hypervisor in selfie, into an operating system with processes, threads and locks, and at each step asking which of the three the code you wrote belongs to.

The purpose is the one shared with the companion classes: a deep understanding of the basic principles, deep enough to position generative AI, and whatever comes next, properly. A kernel is where a system meets things it cannot trust, and that is the shape of the newest problem too.

THE ASSIGNMENTS

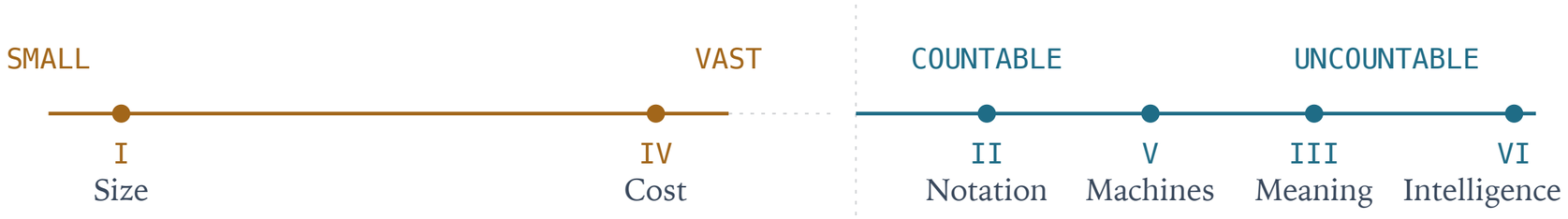
Ten autograded extensions of selfie: an assembler, processes, fork and wait, locks, threads, a thread-safe malloc, a lock-free stack. Plus one new one in week 10: state a memory-safety property for a piece of your systems code and let a model checker find the input that breaks it.

THE BOOK

What is Intelligence? — the Computing chapter is this class; the Meaning and Cost chapters are its theory. Read the Introduction and the Selfie chapter this week.

THE AXIS

One line carries the semester.



the six stations of this book, on its one axis

the six stations, and where a systems class dwells

Countable. The machine, its privilege levels and exceptions, the assembler as a regular language. Weeks 2 and 3: everything still decidable.

Uncountable. Isolation. Memory, time, and the kernel that must isolate itself. Weeks 4 to 9: paging, scheduling, self-hosting, processes, threads, garbage collection — and Gödel, Turing and Rice at each of them.

Cost. Verifying systems code with a solver, and the price of everything in caches, cycles and joules. Weeks 10 and 11. Then universality for systems, agents as processes, and the definition.

THREE COMMANDS, THREE THEOREMS

Understand the third and you understand operating systems.

```
$ ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m && diff -s selfie1.m selfie2.m
$ ./selfie -c selfie.c -o selfie.m -m 2 -l selfie.m -m 1
$ ./selfie -c selfie.c -o selfie.m -m 3 -l selfie.m -y 2 -l selfie.m -y 1
```

SELF-COMPILATION · A FIXED POINT

The compiler class. Here only its lesson: a system that agrees with itself has certified nothing, and the outside check is the rule.

SELF-EXECUTION · THE UNIVERSAL MACHINE

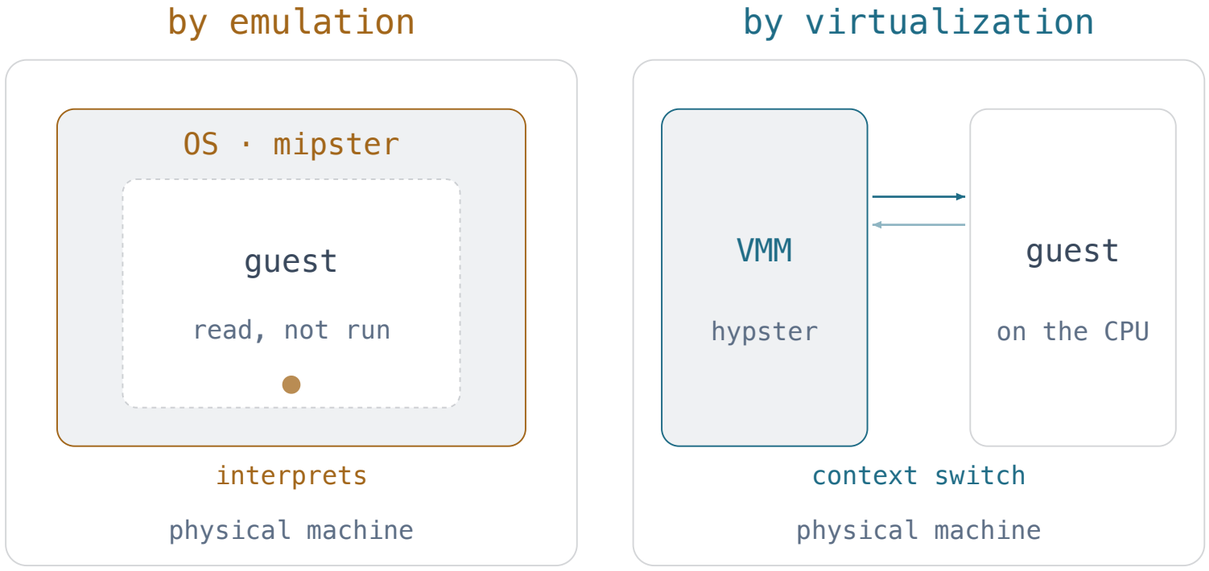
An emulator that runs every program for its machine, itself included. The operating system by emulation, and the executable specification of the one by virtualization.

SELF-HOSTING · THE BOOTSTRAP PROBLEM

A hypervisor whose virtual machines can run the hypervisor. The kernel must be isolated from the things whose isolation it provides. Gödel's second theorem, run as an operating system. This class.

THE THESIS

An OS by emulation is semantically equivalent to an OS by virtualization.



two ways to build one operating system

By emulation the guest is read; by virtualization the guest is run, on the same processor as the kernel. Same guest, same 86,380 instructions, same output. Only the bill for the host differs: ×2,593 against ×208.

So virtualization is needed only for performance, and it buys that performance by introducing self-reference: the kernel needs what it provides. Weeks 4 to 6 build up to that sentence and week 6 lands it.

In my experience this is the one thing whose absence stops students from ever understanding what a kernel is. Most courses show only the virtualized design, with the loop tangled into paging and scheduling and never named. Here it is named first.

FOURTEEN WEEKS

The route.

WK	LECTURE	ASSIGNMENT
2	The machine again: privilege, exceptions, system calls; the assembler	assembler-parser
3	Emulation: mipster as universal machine; the price of interpretation	self-assembler
4	Virtual memory: paging; virtual as notation, physical as meaning	—
5	Time-sharing: the timer interrupt as the answer to halting; scheduling	processes
6	Self-hosting: hypster; the bootstrap problem as Gödel II; the TCB	fork-wait
7	Processes: fork, wait, exit; a process is a virtual machine	fork-wait-exit
8	Concurrency: threads, locks, lr/sc; interleavings as state space	lock, threads
9	Runtime systems: malloc, garbage collection; liveness by reachability	threadsafe-malloc
10	Verifying systems code: rotor, bounded model checking	treiber-stack, rotor-bounds
11	Cost: caches, performance, energy; Landauer; Goodhart	—
12	Universality for systems: halting, Rice on schedulers, deadlock	—
13	Agents as processes: sandboxing is virtualization; the gate	—
14	What a system is; the definition	—

THIS WEEK

Take a selfie.

1. Install selfie, run the three commands, and set up the grader: private clone, upstream, commit links.
2. Do `print-your-name`, prefixed like every other status message, and check it with the grader.
3. Run `make emu`, `make emu-emu` and `make os-emu`, and write down the guest's and the host's instruction counts for each.
4. Read the Introduction, the Selfie chapter, and the Computing chapter's opening callout.

NEXT WEEK

The machine again, from the kernel's side: privilege, exceptions, system calls, machine contexts — and the assembler, which is your first assignment and a regular language.