2023 CCF CHINASOFT 中国软件大会

The 2023 USENIX Annual Technical Conference (USENIX ATC)

"The Hitchhiker's Guide to Operating Systems"

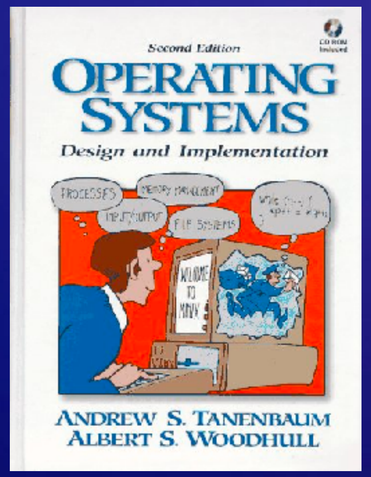
《操作系统》教学中缺失的一角

蒋炎岩

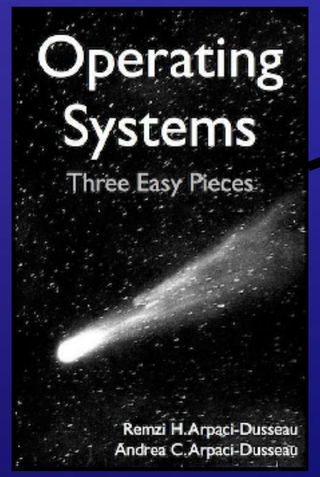
jyy@nju.edu.cn

南京大学

如何向下一代年轻学者介绍“系统人到底在做什么”?



我学操作系统时用的

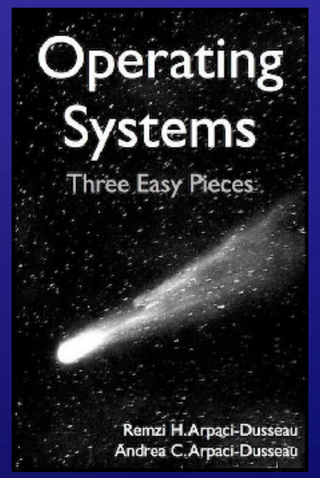


Gap

我教操作系统时用的

我实际日常看的

"There is a body of software, in fact, that is responsible for making it easy to run *programs*…"



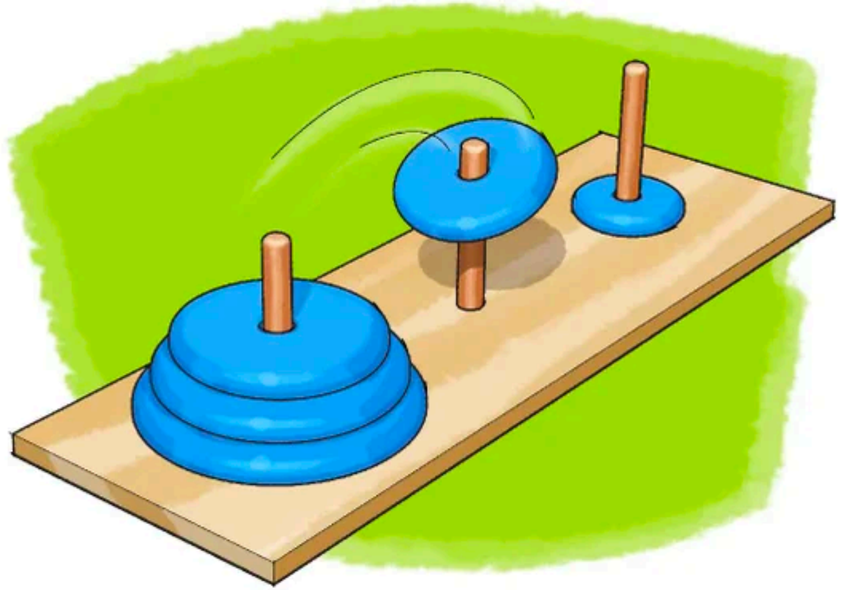
但是……

学生并不知道什么是**程序**……

例子：The Tower of Hanoi

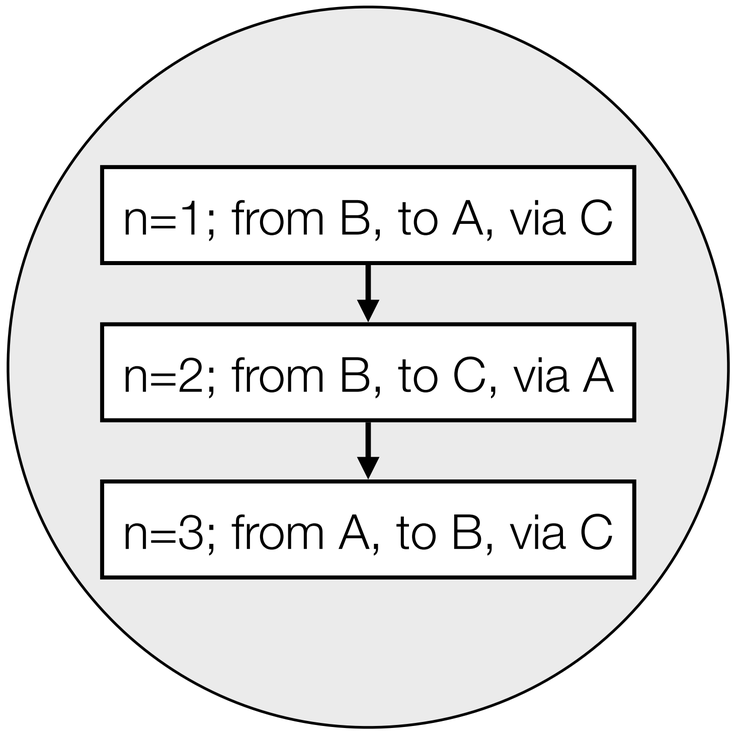
- 还记得被递归支配的恐惧吗?
  - 那么，如果让你写一个非递归……呢?

```c
void hanoi(int n, char from, char to, char via) {
  if (n == 1) {
    printf("%c → %c\n", from, to);
  } else {
    hanoi(n - 1, from, via, to);
    hanoi(1,     from, to,  via);
    hanoi(n - 1, via,  to,  from);
  }
}
```



程序 = 状态机

- 我们可以 “模拟栈”……但到底模拟什么？我们漏了什么？



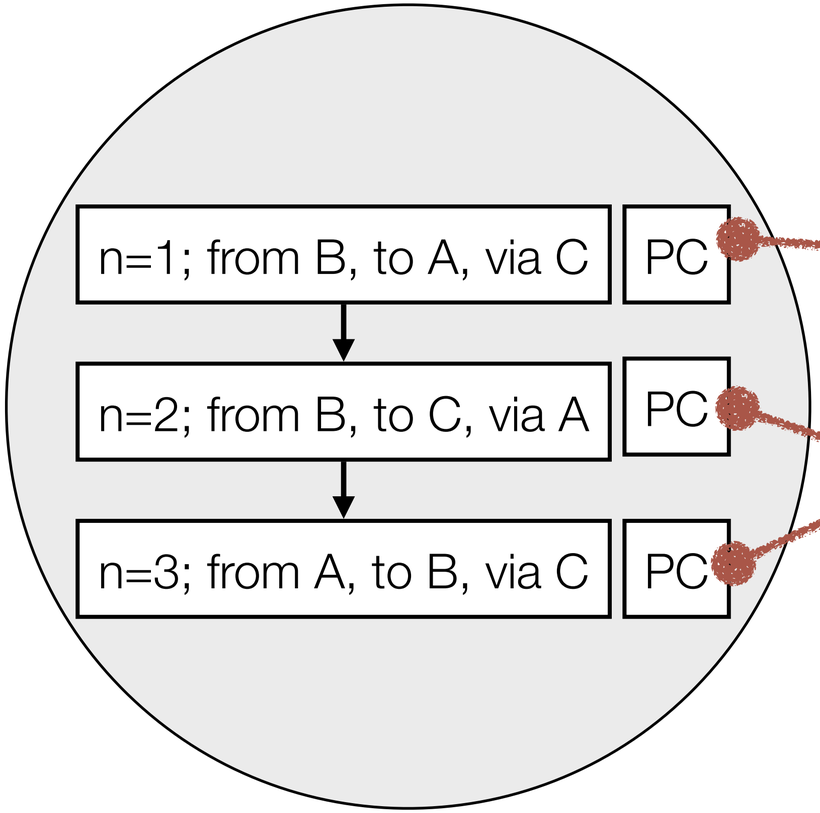
hanoi.c

```
3  void hanoi(int n, char from, char to, char v
4    if (n == 1) {
5      printf("%c -> %c\n", from, to);
6    } else {
7      hanoi(n - 1, from, via, to);
8      hanoi(1,     from, to,  via);
9      hanoi(n - 1, via,  to,  from);
10   }
11 }
12
13 int main() {
14   hanoi(3, 'A', 'B', 'C');
15 }
16
```

```
In: hanoi                                L4    PC: 0x555555555157
ry=67 'C', via=via@entry=66 'B') at hanoi.c:4
#1  0x000055555555518b in hanoi (n=n@entry=2, from=from@e
ntry=65 'A', to=to@entry=67 'C', via=via@entry=66 'B') at
 hanoi.c:8
#2  0x000055555555517a in hanoi (n=n@entry=3, from=from@e
ntry=65 'A', to=to@entry=66 'B', via=via@entry=67 'C') at
 hanoi.c:7
#3  0x00005555555551e0 in main () at hanoi.c:14
(gdb)
```

程序 = 状态机

- 我们漏掉了 Program Counter!




```c
void hanoi(int n, char from, char to, char via) {
  if (n == 1) {
    printf("%c → %c\n", from, to);
  } else {
    hanoi(n - 1, from, via, to);
    hanoi(1,     from, to,  via);
    hanoi(n - 1, via,  to,  from);
  }
}
```

程序 = 状态机


```c
void hanoi(int n, char from, char to, char via) {
  Frame stk[64], *top = stk - 1;
  call(n, from, to, via);
  for (Frame *f; (f = top) >= stk; f->pc++) {
    n = f->n; from = f->from; to = f->to; via = f->via;
    switch (f->pc) {
      case 0: if (n == 1) { printf("%c -> %c\n", from
      case 1: call(n - 1, from, via, to);   break;
      case 2: call(    1, from, to,  via);  break;
      case 3: call(n - 1, via,  to,  from); break;
      case 4: ret();                        break;
      default: assert(
    }
  }
}
```

Top-most frame PC

```c
typedef struct {
  int pc, n;
  char from, to, via;
} Frame;

#define call( ... ) \
  ({ *(++top) = (Frame) { .pc = 0, __VA_ARGS__ };
```

Per-frame program counter

状态机：《操作系统》课程中缺失的一角

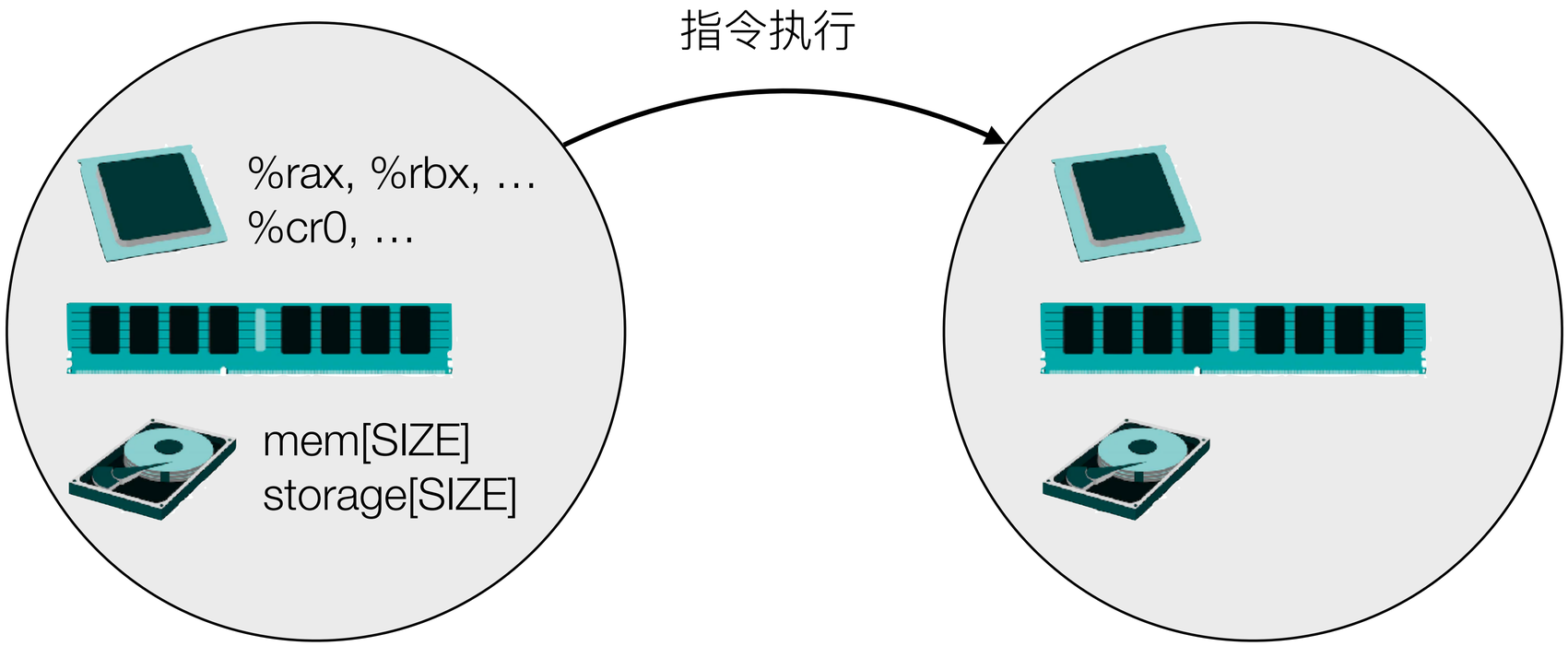
C. Ellison, G. Rosu. An executable formal semantics of C with applications. POPL'12.

Philosophy 1

Everything is a state machine

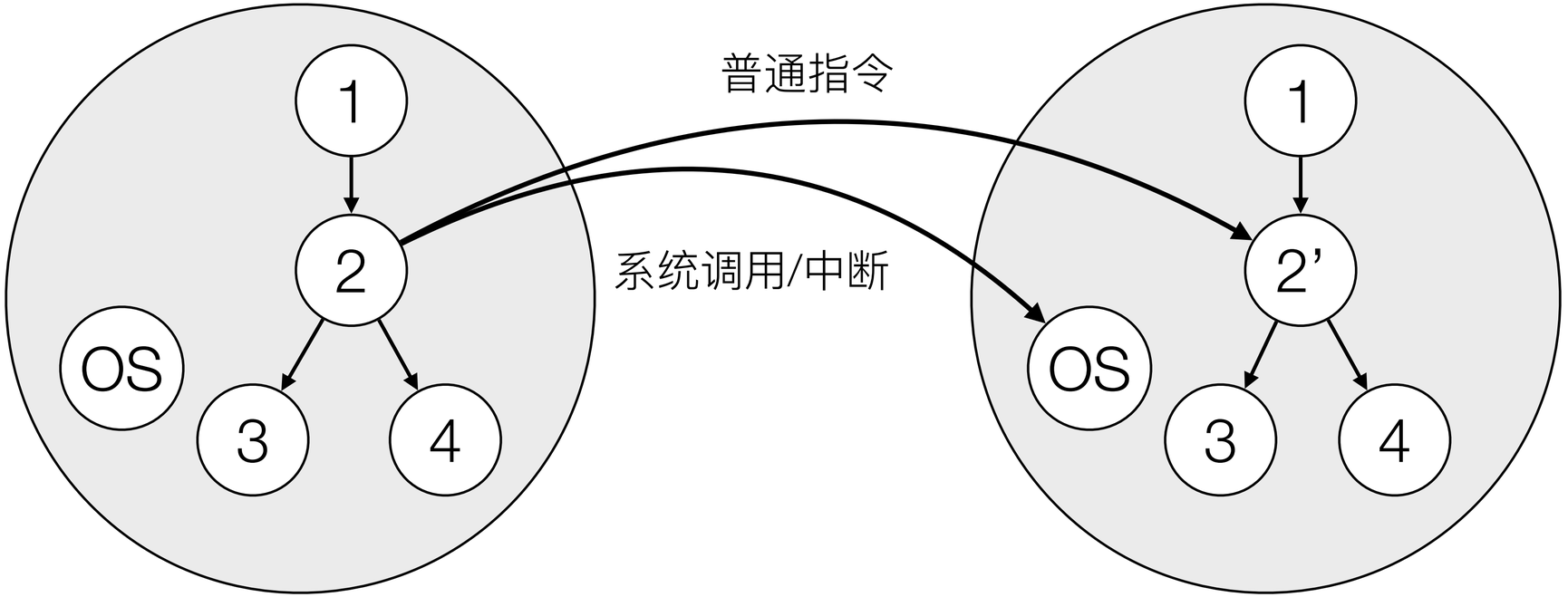
一切都是状态机

- 因为电路是状态机……



那操作系统呢?

- 状态机的 “容器”

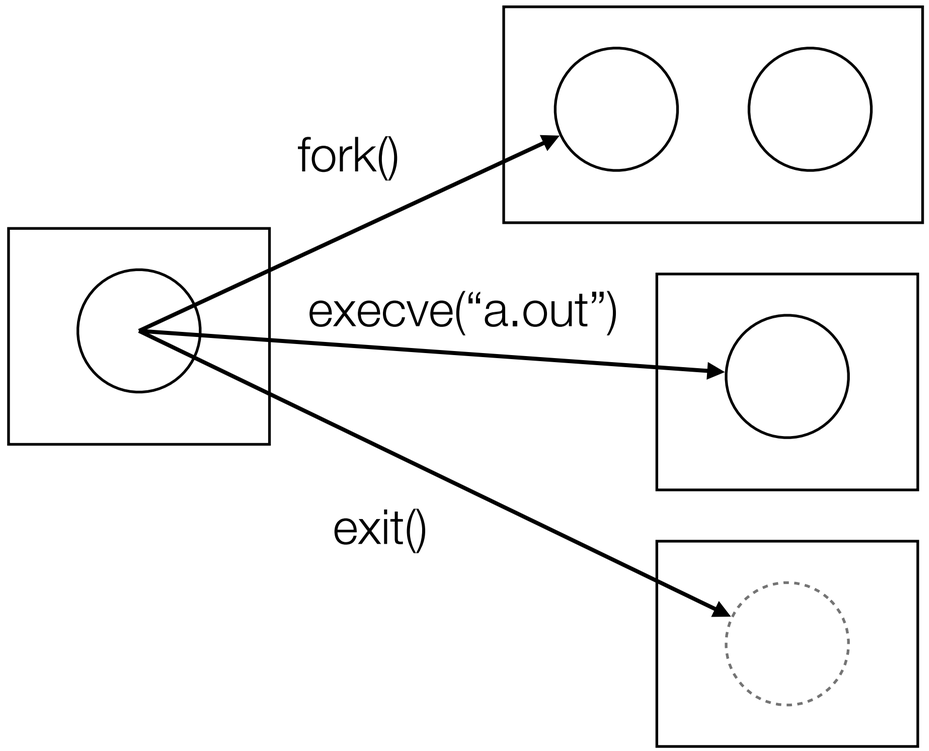


废话……

这有什么实在的好处吗?

Formally, yet informal

《操作系统》课程中的对象有了严格的数学表达



创建状态快照的副本

将当前状态机 “重置” 成 a.out 描述的初始状态

推论：可执行文件 = 状态机初始状态的描述

将状态机从系统中删除

应用：调试一切

0%	0%	0%	0%	0%
0%	0%	0%	0%	0%
0%	0%	0%	0%	0%
0%	0%	0%	100%	0%
0%	0%	0%	0%	0%

在已部署大规模系统中……

SO/GPT: 无效

17

Using Ubuntu 17.04 ISO on a SanDisk 32GB USB stick. My m/board is ASROCK B250M. None of the suggestions I read online worked for me. I noticed, though, that the USB stick was mounted when I pulled it out and plugged it back in, so when the installation was showing the Ubuntu 'wait' screen with flashing dots, I figured it was looking for the installation media, so I pulled out the USB stick and pushed it back in. Hey presto! - the installation media was found and I managed to install Ubuntu.

Share Improve this answer Follow

answered Aug 15, 2017 at 3:54
Rhys Parsons
171 ●1 ●2

3 The same worked for me. I pulled it out as soon as the flashing dots came up. – Michael Mulqueen Sep 14, 2017 at 14:52

2 same working for me, really strange, but only in linux in windows detect automatically – Jonathan Rodriguez Feb 8, 2018 at 20:16

SanDisk 32GB USB stick here, with the same symptoms and same workaround. Perhaps there's something wonky about that particular USB stick. – vegai Aug 26, 2019 at 5:59

SanDisk 32GB here, too. An older one that slides back and forth to reveal either USB-A or microUSB. This solution works for me as well. Too strange... – SeinWecko Dec 17, 2019 at 22:11

I could not believe that this is the right answer :) but it was for me. ASROCK Z68 Pro3 – Tomen Mar 23, 2020 at 11:37

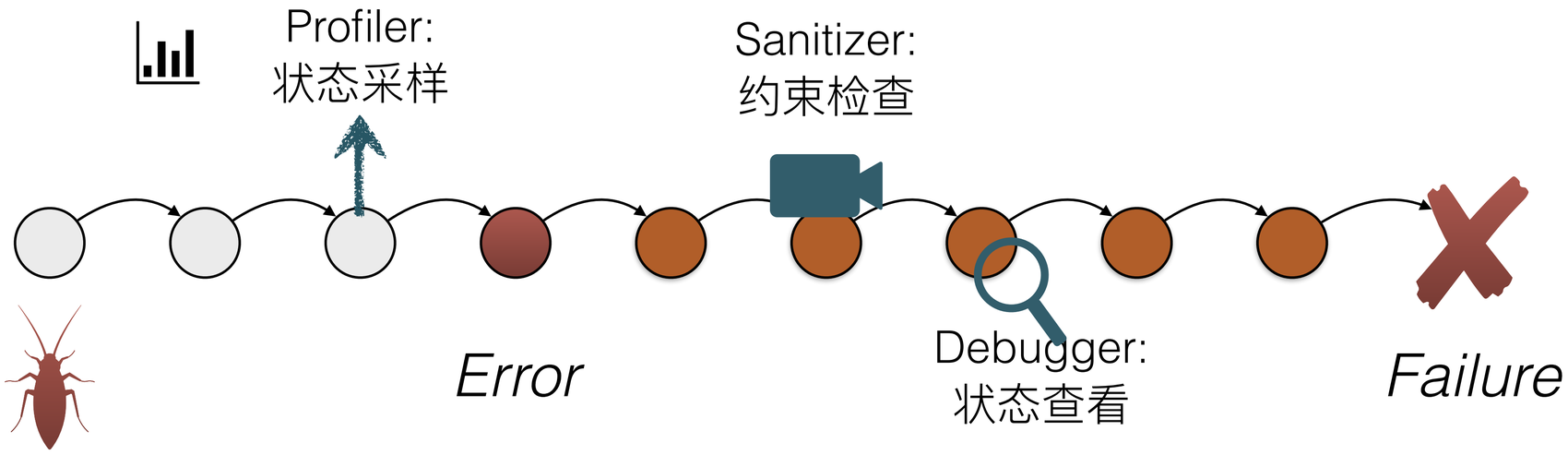
1 Oh thank you! I created an account just to thank you as I'd never have realised this was the problem. Works for me now I've switched to another non-SanDisk USB stick. (For Ubuntu 20.04 the screen where you need to pull out/reinsert now has a rotating logo, instead of flashing dots.) – Stevo Jul 6, 2020 at 12:41

Also have a SanDisk 32GB USB 3.0 stick, same problem, same solution! Thank you!! I spent a couple hours on this over several days before I found this thread. – williamtx Dec 9, 2020 at 8:51

UEFI 能够启动镜像，但
“(initramfs) Unable to find a medium containing a live file system”

如果程序是状态机?

- 一切调试行为 = 状态机 Trace 上的查询



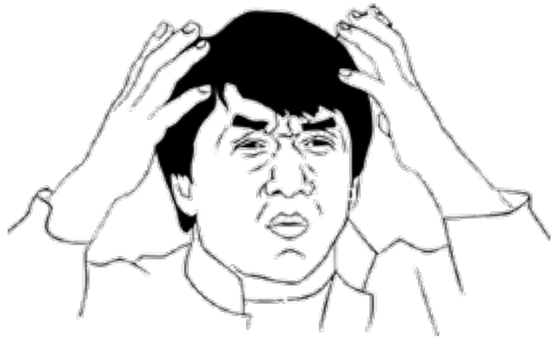
A. Quinn, J. Flinn, M. Cafarella, B. Kasikci. Debugging the OmniTable way. OSDI'22.

那编译器是什么?

- 课堂案例：这是为什么?

```c
void Tsum() {
  for (int i = 0; i < N; i++) {
    sum++;
  }
}

int main() {
  create(Tsum);
  create(Tsum);
  join();
  printf("sum = %ld\n", sum);
}
```



gcc -O0: sum = 119790390
gcc -O1: sum = 100000000
gcc -O2: sum = 200000000

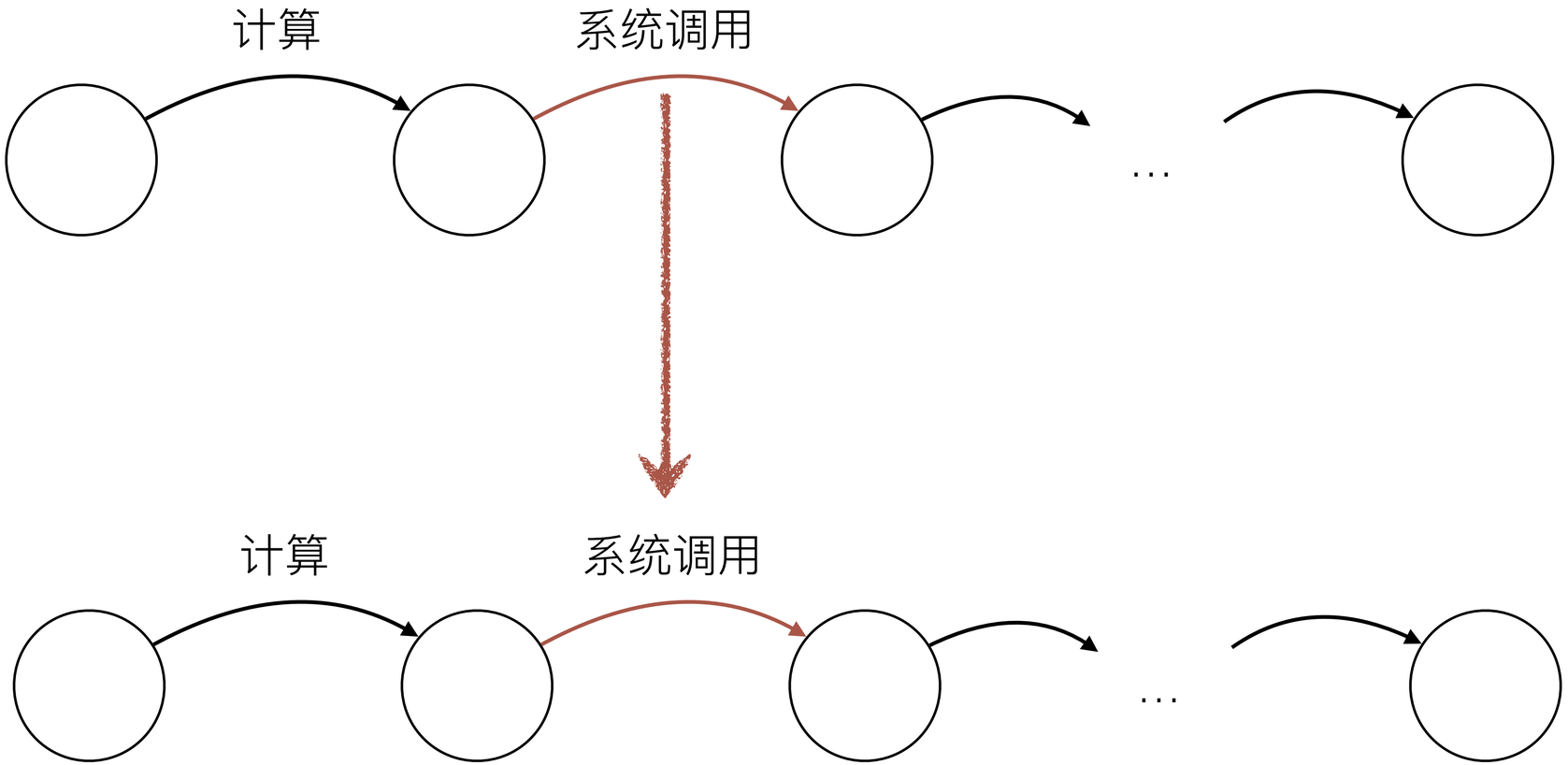
哦，状态机 → 状态机的“翻译”

High-level state machine (C)

```c
void Tsum() {
  for (int i = 0; i < N; i++) {
    sum++;
  }
}
```

Compiler


```
pushq   %rbp
movq    %rsp, %rbp
movq    sum(%rip), %rax
addq    $100000000, (%rax)
popq    %rbp
retq
```



Low-level state machine (assembly)

X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7), 2009.

《操作系统》课教给学生什么？

- 任何时候，如果遇到困难
  - 吸一口气，默念：机器永远是对的；回到状态机吧

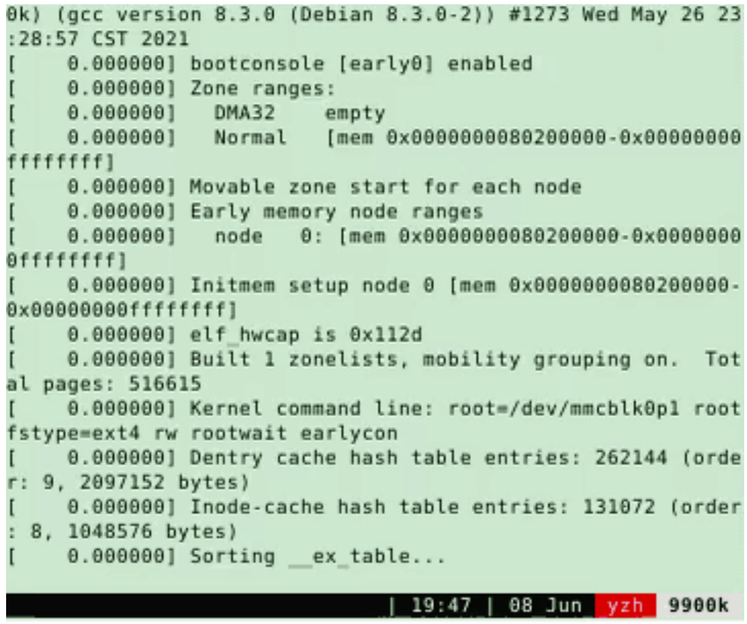
Don’t panic! Computer systems are state machines.

Philosophy 2

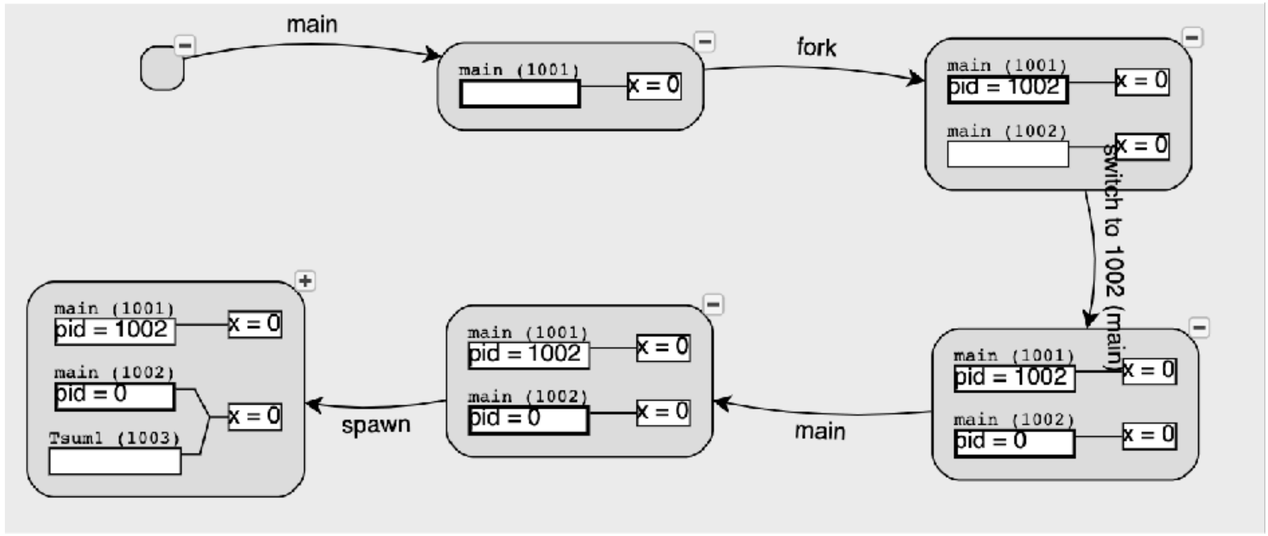
Emulate state machines with executable models

状态机是数学对象

- 我们当然可以模拟它啦 (所以我们可以模拟任何东西)!



软件/硬件全系统模拟器



操作系统上的应用 (模拟系统调用)

是的，你只需要 50 行代码


```
def main():
    sys_write('Hello!')
```

```
yield 'SYS_write', 'Hello!'
```

用 yield 保存状态 (trap)

用 send 实现返回

```
threads = [OperatingSystem.Thread(self._main)]
while threads:  # Any thread lives
    try:
        match (t := threads[0]).step():
            case 'choose', xs:  # Return a random choice
                t.retval = random.choice(xs)
            case 'write', xs:  # Write to debug console
                print(xs, end='')
            case 'spawn', (fn, args):  # Spawn a new thread
                threads += [OperatingSystem.Thread(fn, *args)]
            case 'sched', _:  # Non-deterministic schedule
                random.shuffle(threads)
    except StopIteration:  # A thread terminates
        threads.remove(t)
        random.shuffle(threads)  # sys_sched()
```

```
def step(self):
    """Proceed with the thread until its next trap."""
    syscall, args, *_ = self._func.send(self.retval)
    self.retval = None
    return syscall, args
```

MOSAIC: 10X 代码量

virtualization concurrency persistence

- Processes, threads, and crash consistency!
  - 哦！我们搞定了 OSTEP 上的案例！

System Call	Description
fork()	Create current thread's heap and context clone
spawn(f,xs)	Spawn a heap-sharing thread executing $f(xs)$
sched()	Switch to a non-deterministic thread
choose(xs)	Return a non-deterministic choice among xs
write(xs)	Write strings xs to standard output
bread(k)	Return the value of block k
bwrite(k,v)	Write block k with value v to a buffer
sync()	Persist all outstanding block writes to storage
crash()	Simulate a non-deterministic system crash

应用：A fork() in the road


```c
int main() {
  for (int i = 0; i < 2; i++) {
    fork();
    printf("Hello\n");
  }
}
```

```python
def main():
    heap.buf = ''
    for _ in range(2):
        sys_fork()
        heap.buf += '⭐'
    sys_write(heap.buf)
```

```python
def main():
    for _ in range(2):
        sys_fork()
        sys_write('⭐')
```

A. Baumann, J. Appavoo, O. Krieger, T. Roscoe. A fork() in the road. HotOS'19.

应用：条件变量

cond_wait

```
# release mutex and block
heap.mutex = '✅'
heap.blocked.append(nm)
sys_sched()
while nm in heap.blocked:
    sys_sched()
# reacquire mute
while heap.mutex
    sys_sched()
heap.mutex = '❌'
sys_sched()
```

Atomic
(cannot be interrupted)

cond_broadcast

```
heap.blocked = []
```

Think clean, think rigorous
“数学严格”的《操作系统》课程

upon wakeup

Philosophy 3

Enumeration demystifies operating systems

小小的实现 Trick

- 所有的系统调用都返回 “所有选择”

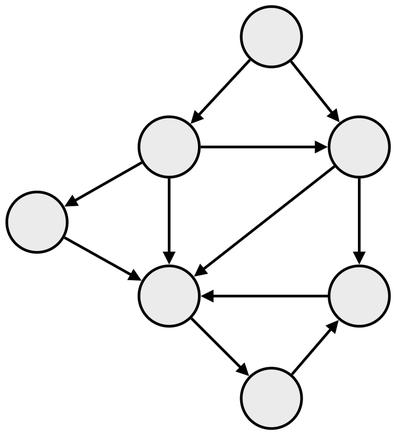
```python
@syscall
def sys_choose(self, choices):
    """Return a non-deterministic value from choices."""
    return {f'choose {c}': (lambda c=c: c) for c in choices}

@syscall
def sys_sched(self):
    """Return a non-deterministic context switch to a runnable thread."""
    return {
        f't{i+1}': (lambda i=i: self._switch_to(i))
            for i, th in enumerate(self._threads)
                if th.context.gi_frame is not None  # thread still alive?
    }
```

买模拟器，白送检查器！


```python
# release mutex and block
heap.mutex = '✅'
heap.blocked.append(nm)
sys_sched()
while nm in heap.blocked:
    sys_sched()
# reacquire mutex
while heap.mutex ≠ '✅':
    sys_sched()
heap.mutex = '❌'
sys_sched()
```

MOSAIC




```
{
  "current": 3,
  "choices": ["t1", "t2", "t3", "t4"],
  "contexts": [
    { "heap": 1, "pc": 4, "locals": { "_": 1 } },
    { "heap": 2, "pc": 4, "locals": { "_": 1 } },
    { "heap": 3, "pc": 3, "locals": { "_": 1 } },
    { "heap": 4, "pc": 4, "locals": { "_": 1 } }
  ],
  "heaps": {
    "1": {}, "2": {}, "3": {}, "4": {}
  },
  "stdout": "🙈🙈",
  "store_persist": {},
  "store_buffer": {},
  "hashcode": "b6b08b3b533b9492",
  "depth": 10
},
```


pipe → grep

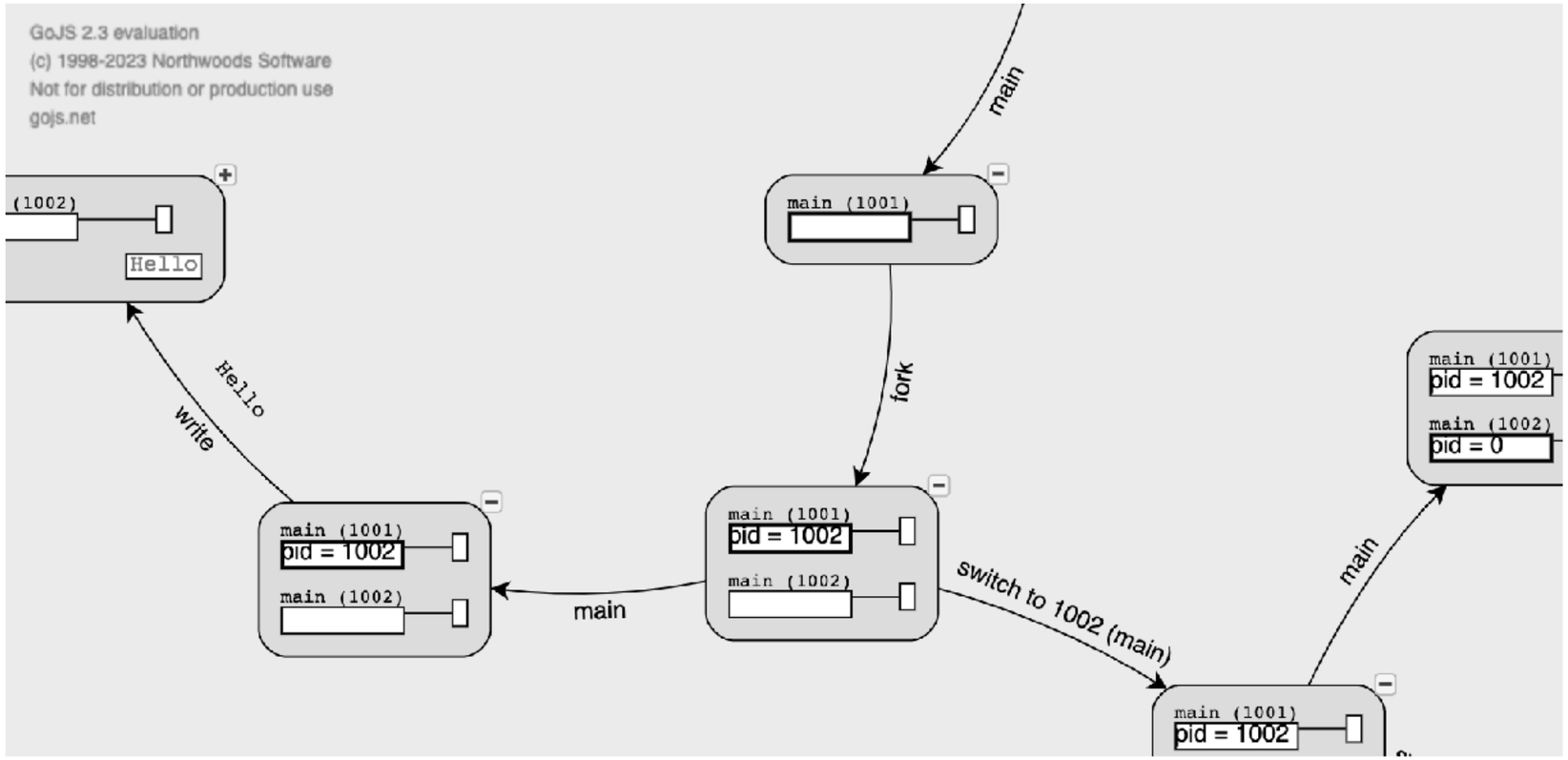
jq

visualization

模型
(Python 代码)

状态迁移图
(JSON)

Prove by brute-force, interactively




```python
def main():
  pid = sys_fork()
  sys_sched()  # non-deterministic context switch
  if pid == 0:
    sys_write('World\n')
  else:
    sys_write('Hello\n')

# Outputs:
# Hello\nWorld
# World\nHello
```

试一试：https://jyywiki.cn/OS/2023/build/lect5.ipynb.html

应用： Crash Consistency Validation

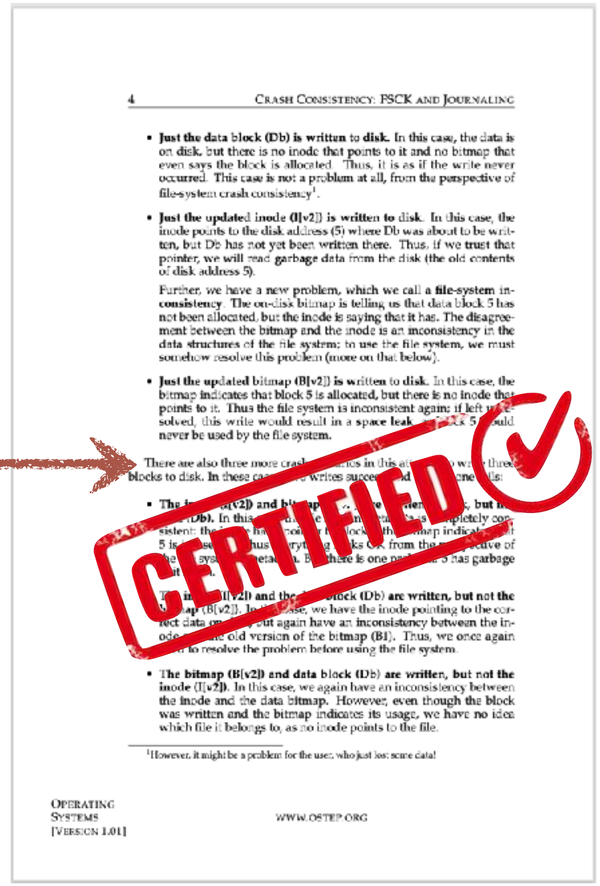

```python
def main():
  # intially, file has a single block #1
  sys_bwrite('file.inode', 'file #1')
  sys_bwrite('used', '#1')
  sys_bwrite('#1', '1-old')
  sys_bwrite('#2', '2-garbage')
  sys_sync()

  # append a block #2 to the file

  sys_bwrite('file.inode', f'file #1 #2')
  sys_bwrite('used', '#1 #2')
  sys_bwrite(f'#2', f'2-new')
  sys_crash()  # system crash

  # display file system state at crash recovery
  inode = sys_bread('file.inode')
  used = sys_bread('used')
  sys_write(f'{inode} | used {used} | ')
  for i in [1, 2]:
    b = sys_bread(f'#{i}')
    sys_write(f'{b} ')
```

```
# Outputs:
# file #1 #2 | used #1 #2 | 1-old 2-garbage
# file #1 #2 | used #1 #2 | 1-old 2-new
# file #1 #2 | used #1 | 1-old 2-garbage
# file #1 #2 | used #1 | 1-old 2-new
# file #1 | used #1 #2 | 1-old 2-garbage
# file #1 | used #1 #2 | 1-old 2-new
# file #1 | used #1 | 1-old 2-garbage
# file #1 | used #1 | 1-old 2-new
```

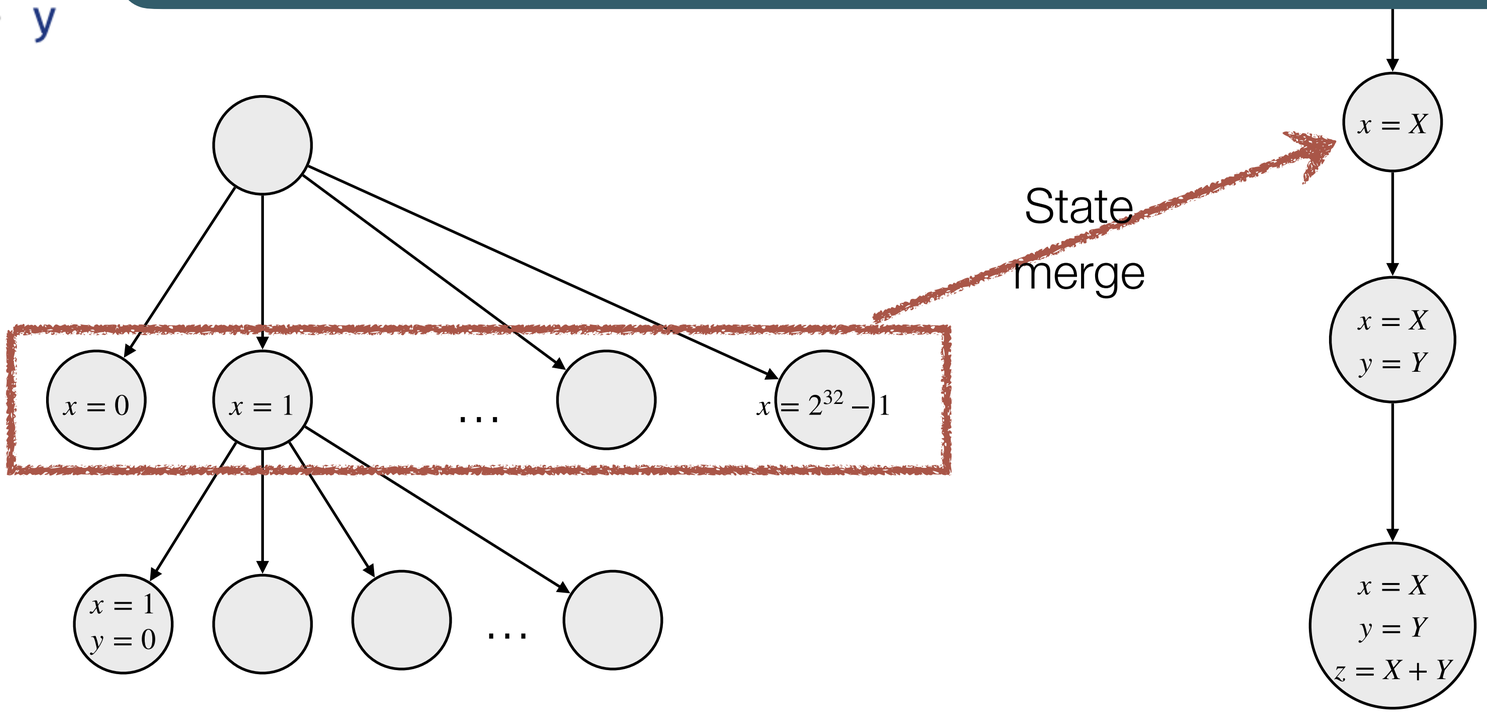


Y. Jiang, H. Chen, F. Qin, et al. Crash consistency validation made easy. FSE'16.

于是，我们可以在 OS 课上尽情发挥

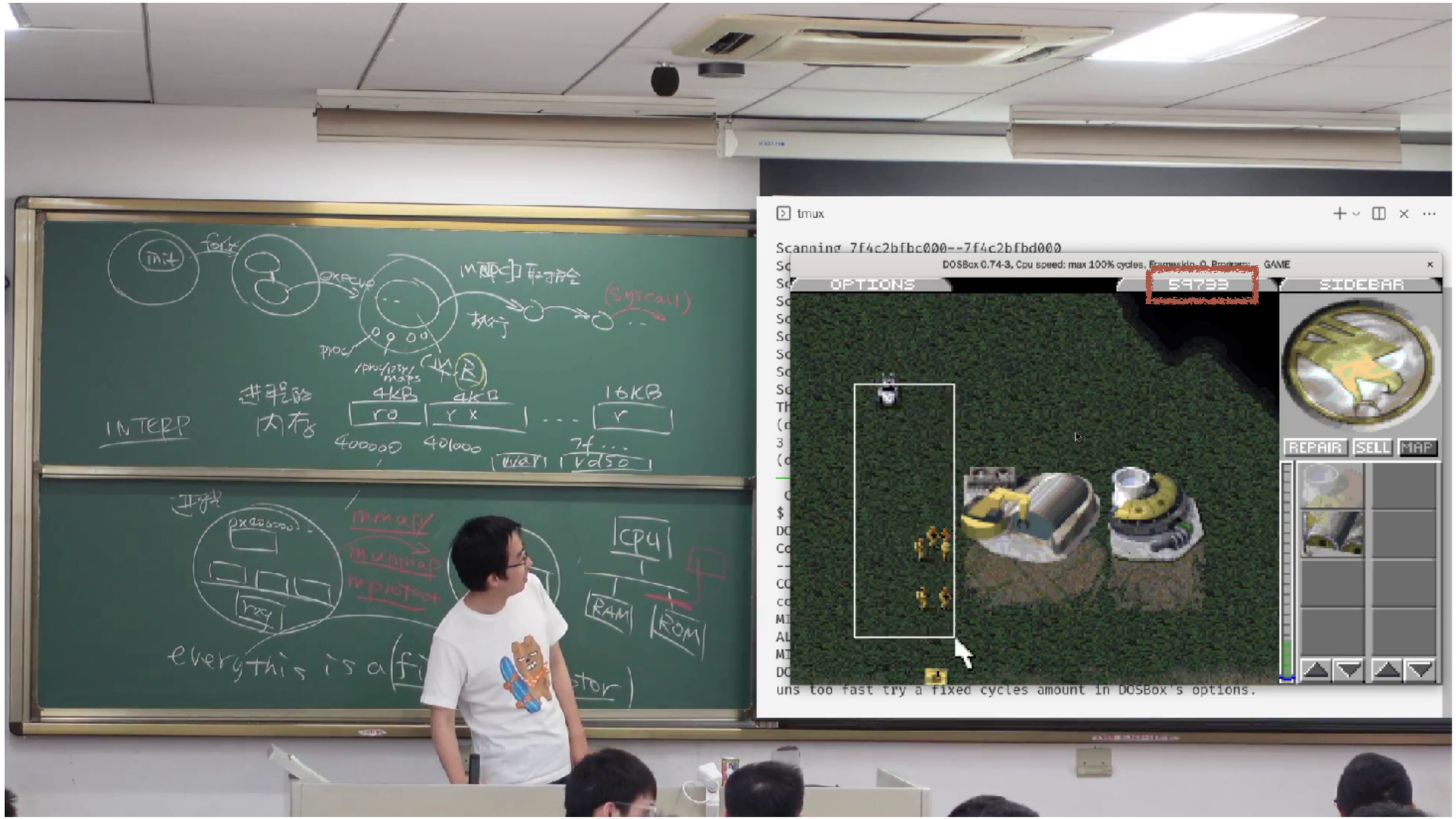
```
# int x = i
x = sys_cho
y = sys_cho
z = x + y
```

枚举是理解一切系统的本质方法

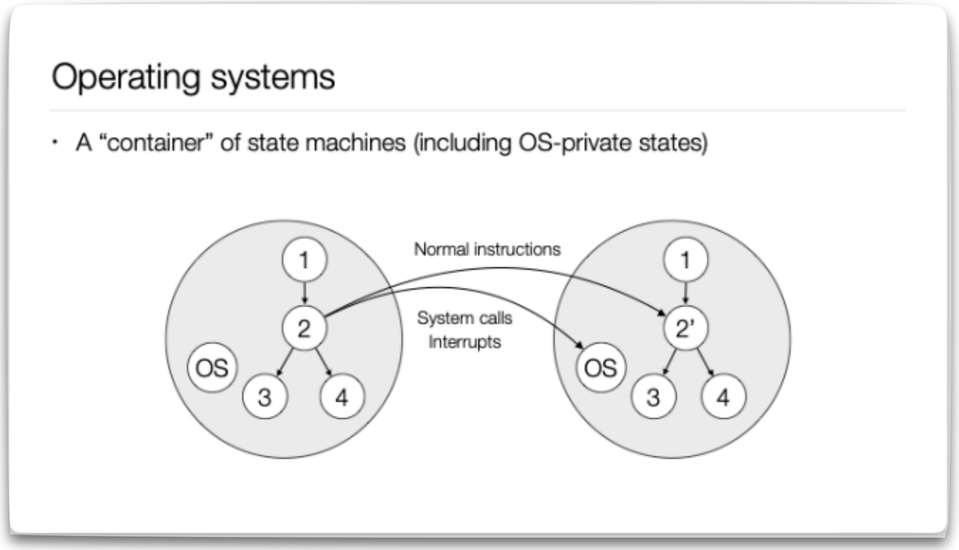


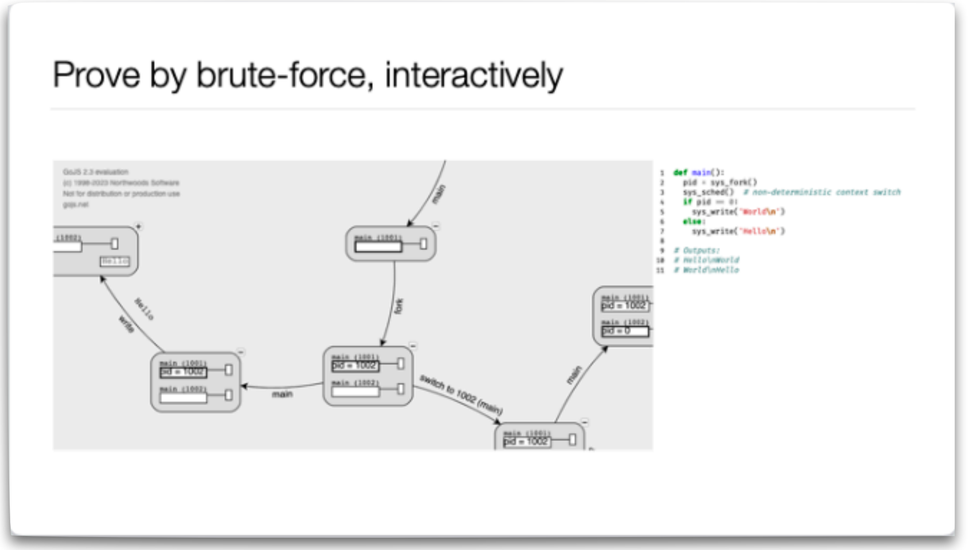
C. Cadar, D. Dunbar, D. Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. OSDI'08.

然后我们有了全新的《操作系统》课！



《操作系统》教学中缺失的一角





Everything is a state machine.

MOSAIC model and checker:

https://github.com/jiangyy/mosaic

课程资料:

https://jyywiki.cn/OS/2023/