The Process Model (1)

L41 Lecture 3, Part 1: The Process Model

Dr Robert N. M. Watson

2020-2021

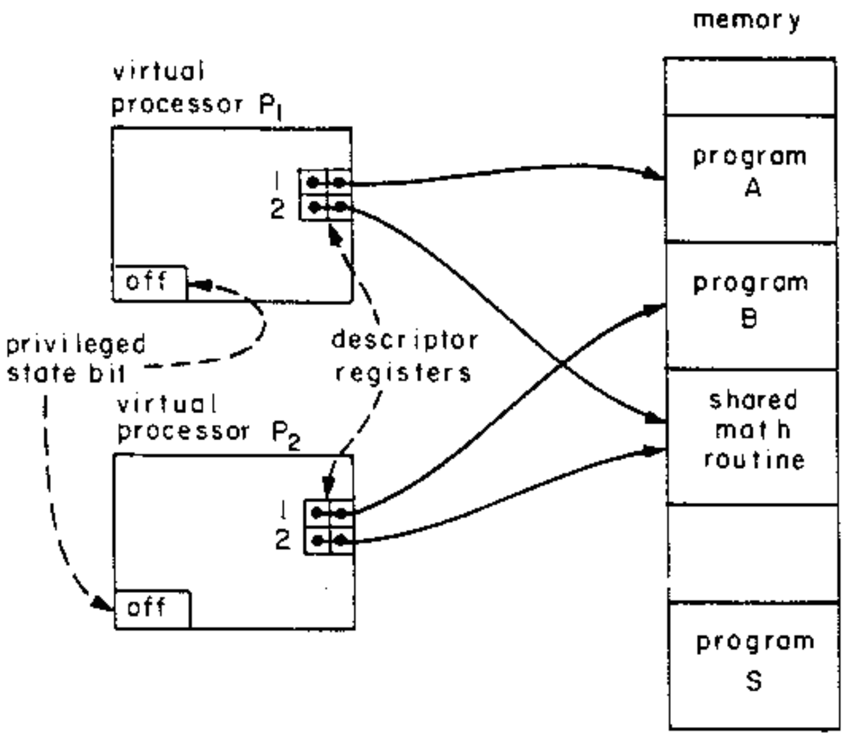
This time: The process model

- The process model and its evolution
- Brutal (re, pre)-introduction to VM
- Where do programs come from?

Lecture 3, Part 1

Lecture 3, Part 2

The **Process Model**: 1970s foundations



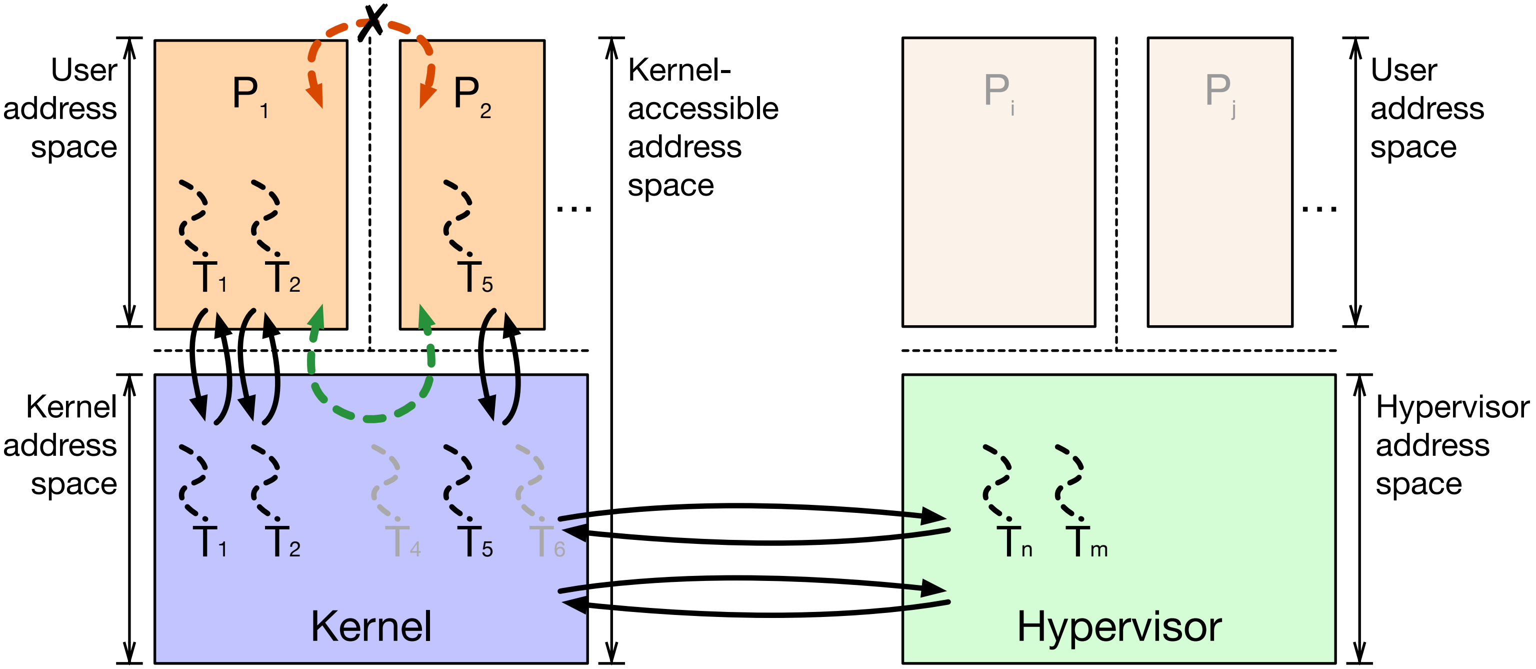
- Saltzer and Schroeder, **The Protection of Information in Computer Systems**, SOSP’73, October 1973. (CACM 1974)
- **Multics process model**
  - ‘Program in execution’
  - **Process isolation** bridged by **controlled communication** via **supervisor** (kernel)
- Hardware foundations
  - Supervisor mode
  - Memory segmentation
  - Trap mechanism
- Hardware protection rings (Schroeder and Saltzer, 1972)

The **process model**: today - concept

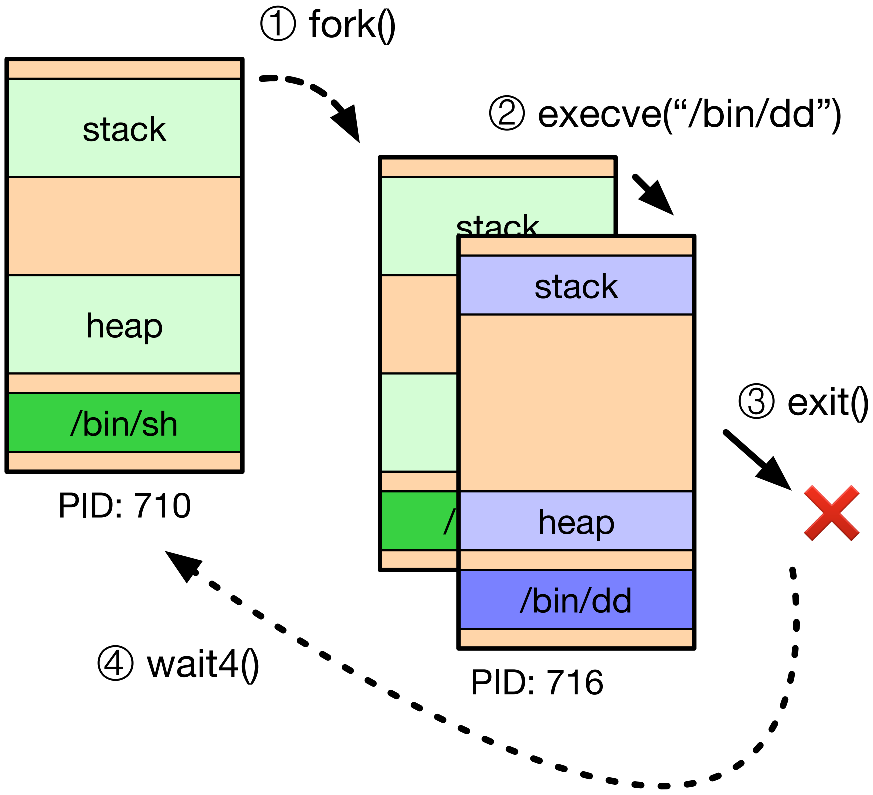
- ‘Program in execution’
  - **Process** ≈ address space
  - **Threads** execute code
  - Unique instance of global variables, etc.
  - Isolated failure domain
- Unit of **resource accounting**
  - Open files, memory, ...
- Unit of privilege
  - Process credentials – UID, OS privileges, MAC, RBAC, ...
  - NB: Increasing support for per-thread credentials
- Recently: Inverted App-OS trust model
  - Third-party applications cannot trust the OS ...
  - E.g., Trustzone, SGX, ...

The **process model** today: isolation and controlled communication

- Hardware foundations for isolation
  - Rings control MMU, I/O, etc.
  - MMU to construct mutually exclusive **virtual address spaces**
  - Context switched **threads of control**
- Hardware foundations for controlled communication
  - Interaction via **traps**: system calls, page faults, ...
  - MMU to construct **shared memory**

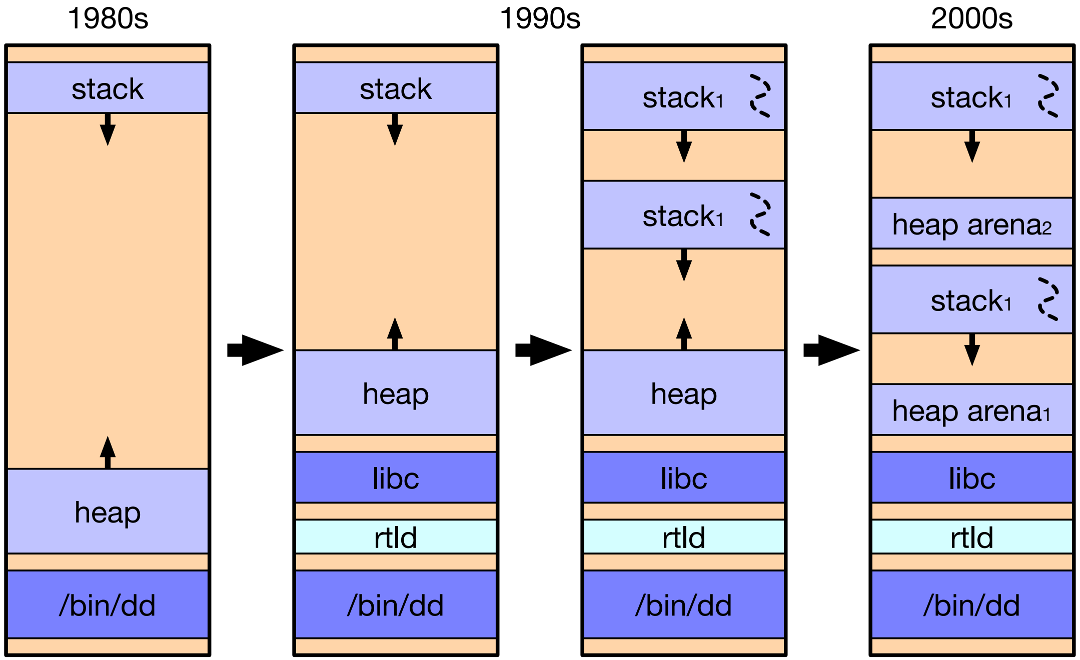


The UNIX process life cycle



- **fork()**
  - Child inherits address space and other properties
  - Program prepares process for new binary (e.g., stdio)
  - Copy-on-Write (COW)
- **execve()**
  - Kernel replaces address space, loads new binary, starts execution
- **exit()**
  - Process can terminate self (or be terminated)
- **wait4()** (et al)
  - Parent can await exit status
- NB: **posix_spawn()**

Evolution of the process model



- **1980s**: Code, heap, and stack
- **1990s**: Dynamic linking, threading
- **2000s**: Scalable memory allocators implement multiple **arenas** (e.g., as in jemalloc)
- Co-evolution with virtual memory (VM) research
  - Acetta, et al: *Mach* microkernel (1986)
  - Navarro, et al: *Superpages* (2002)