Process: Environments and APIs

Pradipta De
pradipta.de@sunykorea.ac.kr
Today’s Topic

• Processes
  – Creation and management
  – Data structures involved

• Later: process scheduling
Process: Definition

• A Process is a program in execution
  – Program ➔ static; Process ➔ dynamic

• Abstraction to time-share the CPU

• Threads and Lightweight Process
  – Abstraction to reduce the overhead of maintaining process
  – Collection of threads belong to a process
  – Older Linux left thread management to user; current version uses the POSIX library and kernel manages scheduling of threads
Process API

• OS gives the APIs to user to manage process

• Process APIs
  – Create: setup necessary data structures to manage the lifetime of a process
  – Destroy: clean up a process
  – Wait: waiting for other process to finish, or I/O operation to complete
  – Status: what is the state of the process, how long has it been running
  – Other controls: suspend
Process States

- **Running**: in execution (task_running)
- **Ready**: waiting to be assigned a processor
  - in one of the runqueues
- **Blocked/Waiting**: waiting for some event to occur
  - I/O to complete or a signal to receive
- **New**: process is being created
- **Terminated**: process is killed
  - Exit_zombie
  - Exit_dead
Process State Transition

new → admitted → ready → waiting → scheduler dispatch → running → exit → terminated

I/O or event completion → waiting

scheduler dispatch → running

I/O or event wait → waiting
Process Control Block
(Process Descriptor)

• Need a data structure to keep track of one process
  – Process control block contains ALL info related to a process
  – In Linux, process descriptor is task_struct

• OS must track all the processes in system
  – Doubly linked list of task_struct
Process Descriptor List

• How to identify a process descriptor?
  – Process descriptors are allocated from slab allocators
  – Can use the address to identify each one
  – Linux also has PID
    • PID for a group of threads is same; stored in tgid
    • Getpid return tgid from thread group
Process Descriptor

- process state
- process number
- program counter
- registers
- memory limits
- list of open files

Logical view
Thread_info structure

• Need to make it easy to find the task_struct of a process
  – Keep it at an address easy to find
  – Prior to 2.6: task_struct stored at the bottom of kernel stack of each process
  – Linux 2.6:
    • slab allocator dynamically assigns the task_struct ➔ cannot assign fixed location
    • Thread_info structure hold pointer to task_struct;
    • Thread_info stored at the bottom of the kernel stack
Context Switching leads to “wasted” CPU cycles
Hardware Supported Context Switch

- Processes share cpu registers, although address space is separate
  - Hence must save the registers (hardware context)
- Older Linux: Used far jmp instruction
  - CPU saves the old hardware context; loads the new context
- Linux 2.6: Software based context switch
  - mov instns to save registers
Implementing process creation

- Processes have parent child relations
  - There is a process with pid 0 – the root of all processes (idle process)

- While creating a child process
  - Duplicate resources owned by a parent for child
    - Copies entire address space of parent
    - Child rarely needs the entire address space;
    - Often wipes out the address space with exec call
  - Current systems uses
    - Copy-on-Write
    - Lightweight process or threads
      - Shares kernel data structure, such as Page table (address space)
    - vFork: shares the memory address space, but blocks parent execution
clone: process creation

• clone(CLONE_VM | CLONE_FS | CLONE_FILES | CLONE_SIGHAND, 0);
  – CLONE_VM: share memory descriptor and page table
  – CLONE_FS: Share the table that identifies root dir and working dir, and file permissions
  – CLONE_FILES: Share open files
  – CLONE_SIGHAND: share the signal handlers

• What does this call create?
  – A thread
Fork, vfork

• Internally fork uses clone with different flags
  – What will be the pid of a parent and child process after fork is called?

• Vfork same effect as fork
  – Page table entries are NOT copied
  – the child executes as the sole thread in the parent’s address space, and the parent is blocked until the child either calls exec() or exits.
Putting It Together

- Process: Definitions, data structures
- Process APIs: features and implementation