UNIX

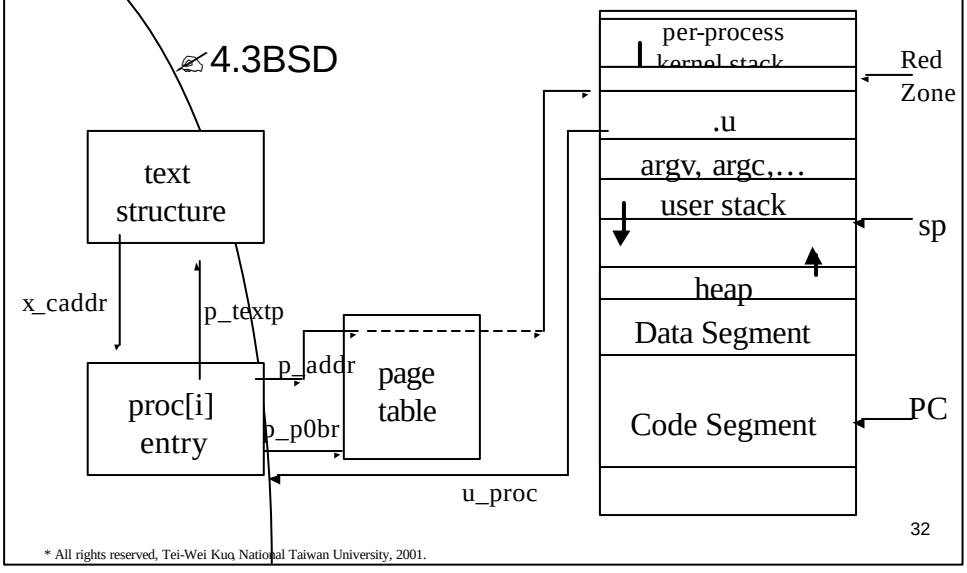
- Introduction
- Programmer Interface
- User Interface
- Process Management
- Memory Management
- File System
- I/O System
- Interprocess Communication

Process Management

- How to represent a process for
  - Process control
  - CPU scheduling
- Process Control Block (PCB)
  - proc[i]
    - Everything the system must know when the process is swapped out.
      - pid, priority, state, timer counters, etc.
  - .u
    - Things the system should know when process is running
      - signal disposition, statistics accounting, files[], etc.

Process Management - DS

- 4.3BSD



32

Process Management - DS

- proc[I] entry
  - pid, ppid
  - user_priority, system_priority
  - state, e.g., SRUN, SSLEEP,, ZOMBIE, etc.
  - signal mask, signal state
  - timer counters
  - Etc
- text structure (memory resident)
  - A list to all processes sharing the text segment – a counter is maintained!

33

Process Management - DS

- Virtual memory address space – user space
  - Text segment
    - Read-only except when debuggers' checkpoints must be set up (rw).
  - Data and stack segments
    - RW mode!
  - .u called user structure
    - System call parameters and return values, table of opened files, etc.

34

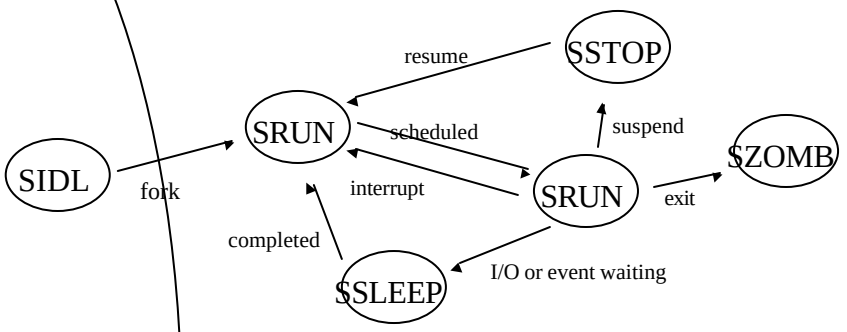
Process Management - DS

- Resources for the process in the kernel space
  - A page table per process
  - per-process kernel stack
    - For the process running in the kernel mode, e.g., for interrupt stacking.
    - System data segment = .u + per-process kernel stack
- Other resources
  - PC, CPU registers, opened files, etc.

35

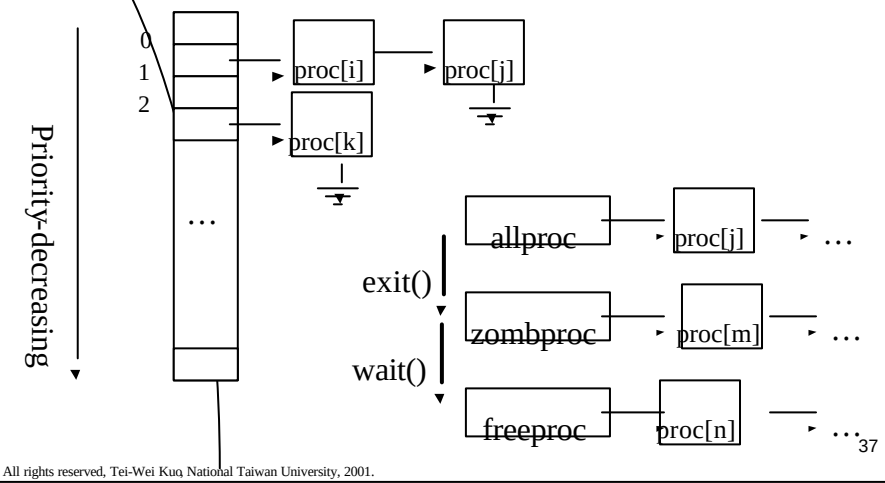
Process Management – Life Cycle

- Process State



Process Management - Lists

- Ready Queue



Process Management – fork

```
if (pid = fork()) {
…
wait();
} else {
execve("a.out");
}
```

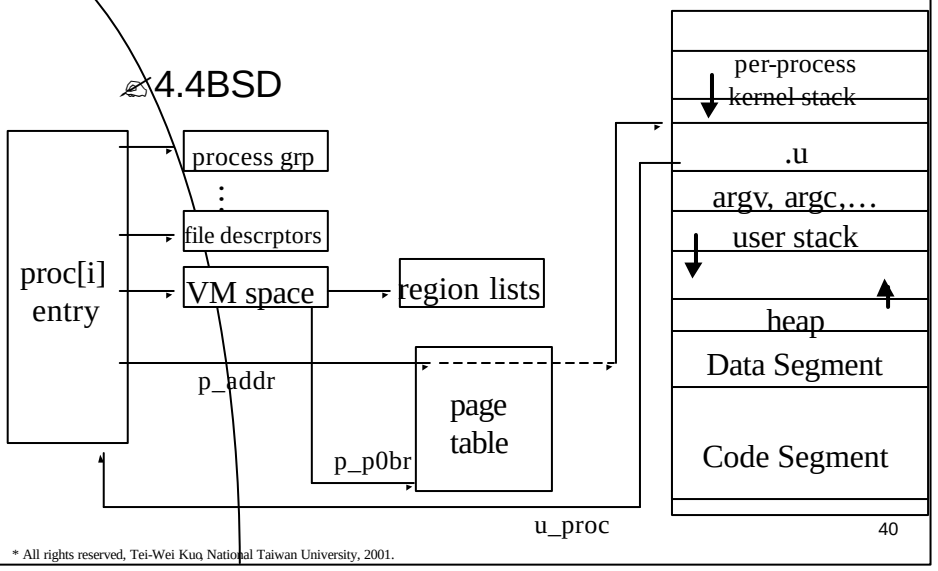
- fork()
  1. Allocate a new proc entry
  2. Register the "text structure"
  3. Allocate memory for data and stack segments
  4. Copy the data and stack segments of its parent to those of the process.
  5. Build a new page table by copying from the page table of its parent!
  6. Copy .u
     - Open file descriptors, usr/grp identifiers, signal handling, etc.

Process Management – execve/vfork

- execve()
  - Discard text, data, and stack segments of the process and reset its page table
  - Load the executable and rebuild text, data, and stack segments and its page table
  - Reset signal handling routines, etc.
- vfork()
  - Borrow segments of its parent
  - Implementation Concerns
    - Suspend the parent until the process terminates or call execve()
    - Or
      - duplicate the page table of its parent
      - Do not create pages unless Copy-on-write pages

Process Management

- 4.4BSD

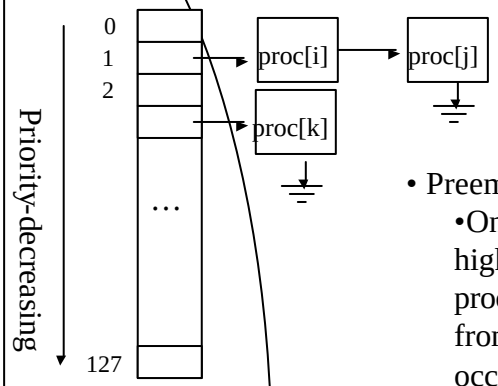


Process Management - Scheduling

- Scheduling Priority
  - User-mode: p_usrpri 50~127
  - Kernel-mode: p_priority 0 ~ 49
    - For the waiting of any event in the kernel mode.
    - Processes with p_priority between (PZERO, PUSER) (i.e., 22 and 50) would be waken up by a signal. (in 4.3BSD, PZERO = 25)
- CPU Scheduling
  - round-robin priority-driven scheduling
    - quantum = 100ms

Process Management - Scheduling

✍Ready Queue



- Preemptive Scheduling Policy
  - Once a process arrives with a higher priority while the running process is in the user mode or exits from a system call, a context switch occurs immediately!

42

Process Management - Scheduling

✍ Scheduling Priority

✍Raising Priority

✍Longer period being blocked, e.g., one sec

✍Blocking in the kernel mode

✍nice() ~ -20

✍Lowering priority

✍Recent CPU usage

✍Exit from the kernel mode

✍nice() ~ +20

p_usrpri = PUSER + ceiling(p_cpu/4) + 2p_nice ; every tick
p_cpu = [2load/(2load+1)]*p_cpu+p_nice ; every second
p_cpu = p_cpu [2load/(2load+1)]$^{p\_slptime}$; once the process is awaken.

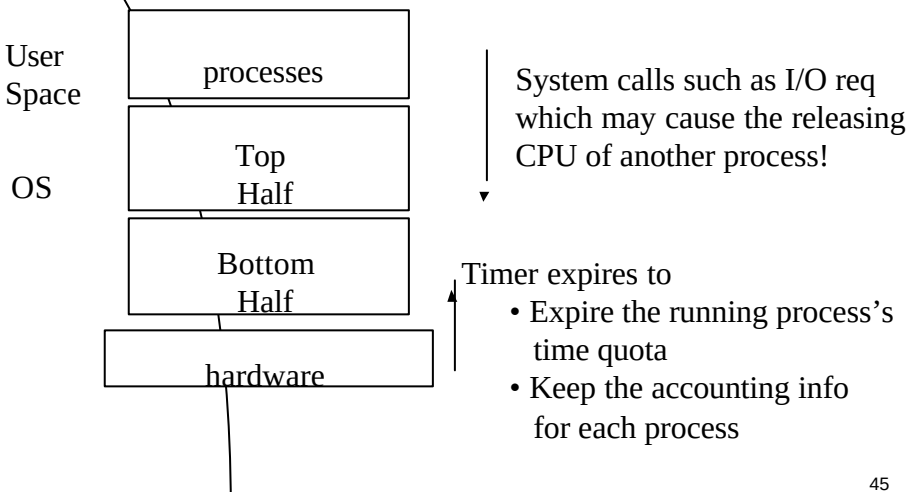
43

Process Management - Scheduling

- Context Switch
  - Synchronous CS
    - Voluntary
      - Call system calls and then sleep
    - Involuntary
      - Time quantum is up!
  - Asynchronous CS
    - Device interrupts

44

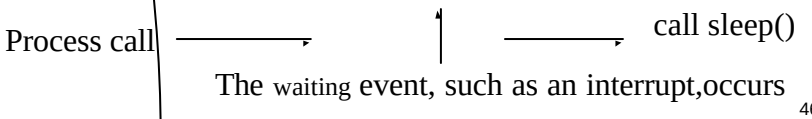
CPU Scheduling - Revisiting



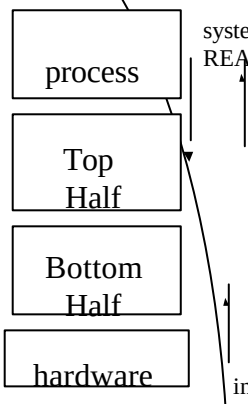
45

Process Management - Scheduling

- Synchronous Voluntary Context Switch
  - A system call finally invoke sleep(&wchan)!
    - wchan – address of some data struture
      - Lbolt: wait for one second
      - proc: wait for child process
      - u: wait for a signal
      - buffer header: wait for I/O operations
        - File reading, block flushing, page fault, etc.
- Race Condition



Process Management - Scheduling



- A general solution in UNIX for resolving race conditions!
  - Raise hardware processor priority
    - e.g., mask interrupts
  - Single-thread kernel
- An obstacle for multi-CPU UNIX!

Process Management

- scheduler (pid = 0)
  - CPU scheduling
- init (pid = 1)
  - Create daemons, login processes, etc.
- pagedaemon (pid = 2)
  - Swapper – mid-term scheduler

48