

Processes - A System View

Concurrency & Context Switching

Process Control Block

What's in it and why? How is it used? Who sees it?

5 State Process Model

State Labels. Causes of State Transitions. Impossible Transitions.

Zombies and Orphans

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

On a single CPU system...

- Only one process can use the CPU at a time
 - ... But we want the appearance of every process running at the same time
- How can we manage CPU usage?
 - "Resource Management"

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



On a single CPU system...

- Your process is currently using the CPU
- What is the O/S (most likely) doing?

```
long count = 0;
while (count >= 0)
  count ++;
```

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



On a single CPU system...

- Answer:
 - Nothing.
- Naively... Put the current process on 'pause'
- What are our options?

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



[O/S : I need the CPU]

1. Time slicing
 - Use a HW timer to generate a HW interrupt
2. Multiprogramming
 - Wait until the process issues a system call
 - e.g., I/O request
3. Cooperative Multitasking
 - Let the user process yield the CPU

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



[Time Slicing]

- A Process loses the CPU when its time quanta has expired
- Advantages?
- Disadvantages?

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



[Multiprogramming]

- Wait until system call
- Advantages?
- Disadvantages?

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



[Cooperative Multitasking]

- Wait until the process gives up the CPU
 - Advantages?
 - Disadvantages?
- ```

long count = 0;
while(count >=0) {
 count ++;
 if(count % 10000 == 0)
 yield();
}

```

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## Context Switch: In a simple O/S (no virtual memory)

- Context switch
  - The act of removing one process from the running state and replacing it with another

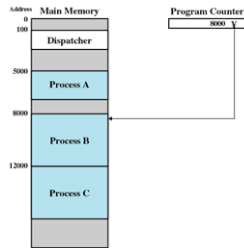


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13  
Copyright © Nahrstedt, Angrave, Abdelzaher, Kravets, Gupta



## Context Switch

- Overhead to re-assign CPU to another user process
- What activities are required?

Copyright © Nahrstedt, Angrave, Abdelzaher, Kravets, Gupta



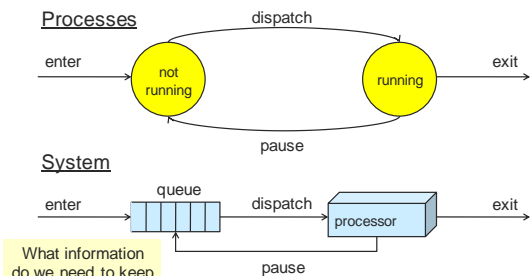
## Context Switch

- Overhead to re-assign CPU to another user process
  - Capture state of the user's processes so that we can restart it later (CPU Registers)
  - Queue Management
  - Accounting
  - Scheduler chooses next process
  - Run next process

Copyright © Nahrstedt, Angrave, Abdelzaher, Kravets, Gupta



## 2 State Model



Copyright © Nahrstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ Process Control Block (PCB) ]

- In-memory system structure
  - User processes cannot access it
  - Identifiers
    - pid & ppid
  - Processor State Information
    - User-visible registers, control and status, stack
  - Scheduling information
    - Process state, priority, ..., waiting for event info

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ PCB (more) ]

- Inter-process communication
  - Signals
- Privileges
  - CPU instructions, memory
- Memory Management
  - Segments, VM control 'page tables'
- Resource Ownership and utilization

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ Five State Process Model ]

- "All models are wrong. Some Models are Useful"
  - George Box, Statistician
- 2 state model
  - Too simplistic
  - What does "Not Running" mean?
- 7 state model
  - Considers suspending process to disk
  - See Stallings 3.2

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



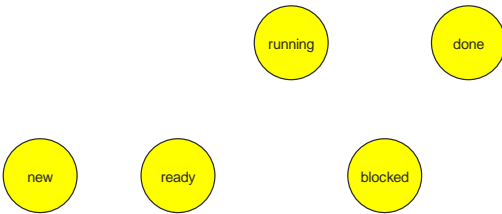
## [ Five State Process Model ]

- Running
  - Currently executing
  - On a single processor machine, at most one process in the "running" state
- Ready
  - Prepared to execute
- Blocked
  - Waiting on some event
- New
  - Created, but not loaded into memory
- Done
  - Released from pool of executing processes

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - States ]

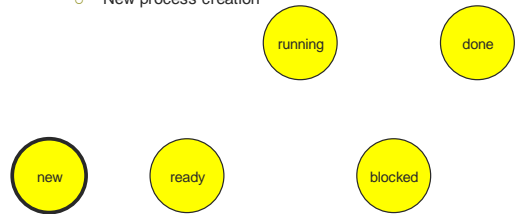


Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]

- Null (nothing) to New
  - New process creation

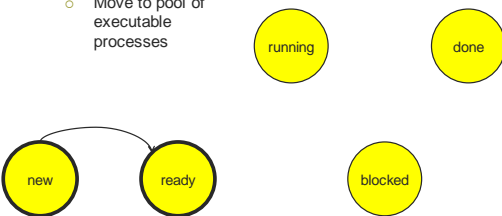


Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]

- New to Ready
  - Move to pool of executable processes

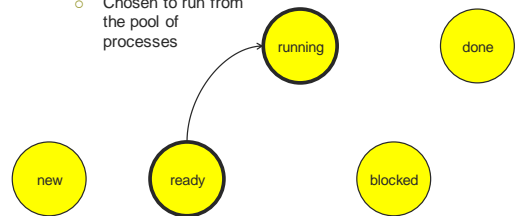


Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]

- Ready to Running
  - Chosen to run from the pool of processes

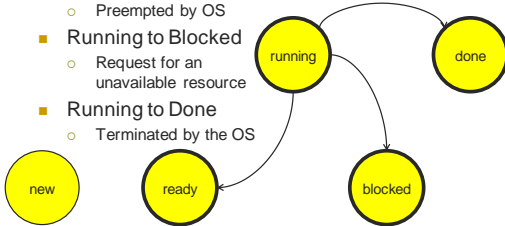


Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]

- Running to Ready
  - Preempted by OS
- Running to Blocked
  - Request for an unavailable resource
- Running to Done
  - Terminated by the OS

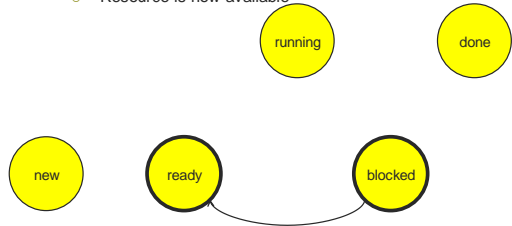


Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]

- Blocked to Ready
  - Resource is now available

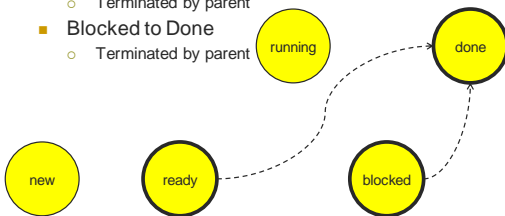


Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]

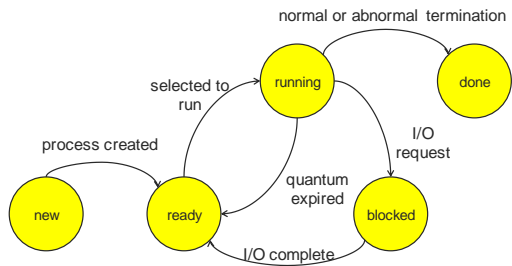
- Ready to Done
  - Terminated by parent
- Blocked to Done
  - Terminated by parent



Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ 5 State Model - Transitions ]



Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ Orphans and Zombies ]

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ Orphans ]

- If the parent process dies no one is left to take care of the child
  - Child may consume large amounts of resources (CPU, File I/O)
  - Child Process is re-parented to the init process
    - init does not kill child but will wait for it.
    - child continues to run and run...

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ Zombies ]

- A Zombie is a child process that exited before it's parent called `wait()` to get the child's exit status
  - Do not consume many resources
    - Exit status (held in the program control block)
  - Also adopted by the init process
- Zombie Removal
  - Professional code installs signal handler (CS241 later lecture) for signal SIGCHLD which issues a `wait()` call

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta



## [ Take-away questions ]

- What would happen if user processes were allowed to disable interrupts?
- In a single CPU system what is the maximum number of processes that can be in the running state?
- Next: Thread Magic

Copyright © Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

