8/25/10

Unix Tutorial / Nachos Install

(36)	SAL 109	Tuesday
(30)	SAL 126	5:30 - 7:00
		7:00 - 8:30
	SAL 109	Wednesday
		5:30 - 7:00
		7:00 - 8:30

Simple O.S.

uniprogrammed

- 1 user at a time
- 1 program at a time

Ex: DOS, Nachos

Result: No security

Sharing the Computer

multiprogrammed

Multiple users

Multiple programs

Need security between user programs

O.S. is responsible for security

O.S. key objective: Keep as many resources on the computer as busy as possible

To do this: The O.S. must switch efficiently between user programs.

⇓

context switch

We are switching the CPU from running one user program to running a different user program

Context switching must be transparent to the user program

When to context switch?

- When a user program requests a "slow" operation
- * When a user program finishes

optional {
- When a new user program arrives
- When a maximum amount of time of occurred ⇒ Time Slice *

Context switching allows for <u>Currency</u>

⇙

User programs can behave as if they have the computer to themself

Result: On a context switch, the *OS* must "remember" the user program context (state) when being evicted from the CPU.

We will use the concept of Process

OSes manage processes, NOT user programs

The process is used to keep track of all the "things" an OS must remember about the state of a user program

3 Main Parts to a Process

1. Code/Data
2. Allocated resources
3. Book keeping information for contextswitch
   - CPU registers
   - Other OS-specific data

Processes Have 4 States

New : Process just been created. Not completely setup for CPU execution, yet, as a user program

Ready: Process is ready for the CPU; waiting its turn

Running: Process currently executing in the CPU.

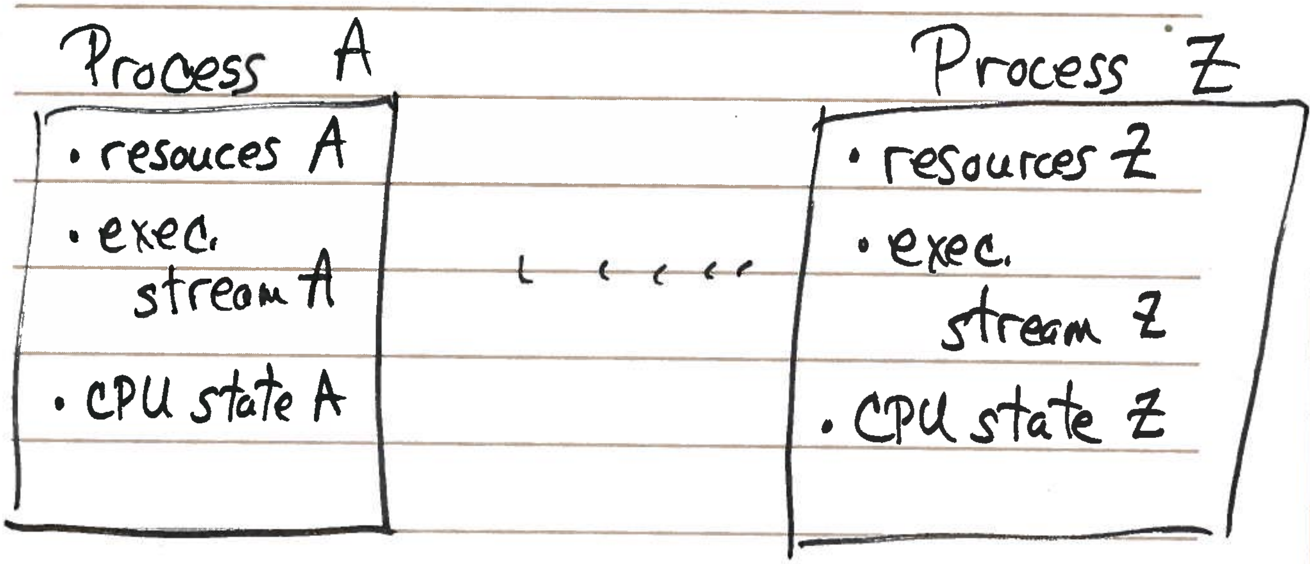
Blocked: Process cannot use the CPU until some event occurs

Processes have a "problem" (based on our design)

Processes have 2 "internal actions"

1. Executing code
   - single execution stream
2. Grouping of resources for access by programs

Current process design only allows for a single execution stream



Sharing of resources can cause problems

⇓

Race Condition

Definition: Order of execution affects the results

Example

```
int i;        // i is shared
```

Process A	Process B
① i = 0;	③ i = 10;
② i = i + 1;	④ i = i - 1;
① → ②	③ → ④

1, 2, 3, 4	9
3, 4, 1, 2	1
1, 3, 2, 4	10
3, 1, 4, 2	0