CS323 Operating Systems
Process

Yuanyuan Zhou
Lecture 3
1/27/2003

Content of this lecture

- Administrative announcements
- What is a process?
- Information in a process
- Summary

Administrative

- Nacho tutorial tonight, same room, 7pm (double check the newsgroup)
- Test quiz available
- Read Chapter 2
  - Wednesday: threads
  - Friday to next Wed: CPU Scheduling
  - Next Friday: Process Synchronization

Mid-term Reschedule (conflict with 321)

1/27-1/31, 5pm CDT	Quiz0 (test quiz)	Sorry, no more future reschedule!!
2/3-2/7,5pm CDT	Quiz1	
2/24-2/28, 5pm CDT	Quiz2	
3/10, Monday, 7-7:50pm, 1320 & 1310 DCL	Mid-exam1	No class on 3/10 (9-10am)
3/17-3/21, 5pm CDT	Quiz3	
4/7-4/11,5pm CDT	Quiz4	
4/21, Monday, 7-7:50pm, 1320 & 1310 DCL	Mid-exam2	No class on 4/21 (9-10am)
4/28-5/2 5pm CDT	Quiz5	
5/10, 11:59pm CDT	Final-exam	

OS: A Friendly Deception

- A simple extended machine
- Manage resources
- Drawback? Maybe lots of extra work
- Benefit: makes life easier for user

Users, Programs, Processes

- Users have accounts on the system
- Users launch programs
  - Many users may launch same program
  - One user may launch many instances of the same program
- Processes: an executing program

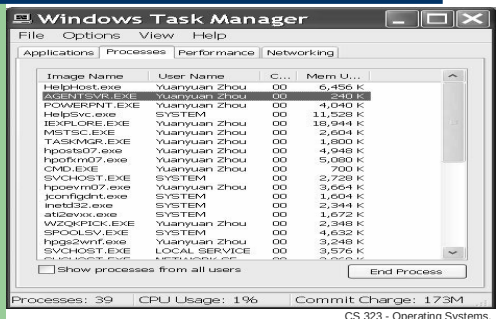
Analogy

- Program: steps for attending the lecture
  - Step1: walk to DCL
  - Step2: enter 1320
  - Step3: find a seat
  - Step4: listen and take notes (or sleep)
- Process: attending the lecture
  - Action
  - You are all in the middle of a process

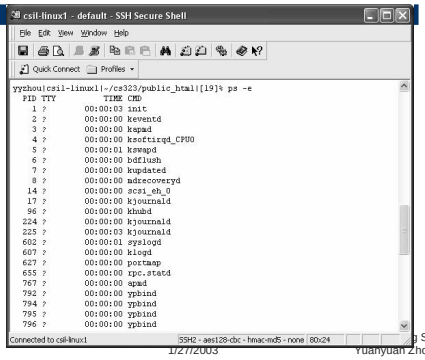
Programs As Process Collections

- Netscape (output of “ps x”)
  - 7253 p0 S 0:19.26 /usr/local/lib/netscape/communicator-4.75.bin -irix-s
  - 7280 p0 I 0:00.15 (dns helper) (communicator-4.7)
- gcc (via “gcc –pipe –v”)
  - /usr/libexec/cpp |
  - /usr/libexec/cc1 |
  - /usr/libexec/as, followed by
  - /usr/libexec/elf/ld

Windows Task Manager



Unix Example: ps



Process Creation

- System initialization
  - reboot
- Execution of a process creation system call
  - Fork()
- User request to create a new process
  - Command line or click an icon
- Initiation of a batch job
  - cron

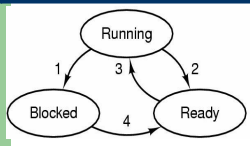
Process Termination

- Normal exit (voluntary)
  - End of main()
- Error exit (voluntary)
  - exit(2)
- Fatal error (involuntary)
  - Divide by 0, core dump
- Killed by another process (involuntary)
  - Kill procID, end task

Process Hierarchies

- Parent creates a child process, a child process can create its own processes
- Forms a hierarchy
  - UNIX calls this a "process group"
- Windows has no concept of process hierarchy
  - all processes are created equal

Process States



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

- Possible process states
  - Running (occupy CPU)
  - Blocked
  - Ready (does not occupy CPU)
  - Other states: suspended, terminated
- Transitions between states
- Question: in a single processor machine, how many process can be in running state?

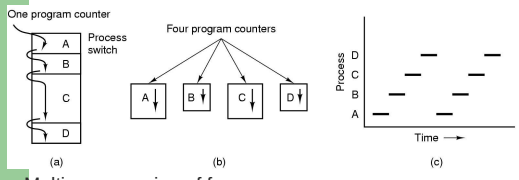
Group Discussion (2 minutes)

- Using "working on a MP" as an example
  - What corresponds to "running"?
  - What corresponds to "blocked"?
  - What corresponds to "ready"?
- Why not the following transitions
  - Ready to blocked
  - Blocked to running

So What Is A Process?

- It's one executing instance of a "program"
- It's separate from other instances
- It can start ("launch") other processes
- It can be launched by them

The Process Model



- Multiprogramming of four programs
- Conceptual model of 4 independent, sequential processes
- Only one program active at any instant
- Real life analogy?
  - A daycare teacher of 4 infants

So What's In A Process? And Why?

- Process State
  - new, ready, running, waiting, halted;
- Program Counter
  - the address of the next instruction to be executed for this process;
- CPU Registers
  - index registers, stack pointers, general purpose registers;
- CPU Scheduling Information
  - process priority and pointer;
- Memory Management Information
  - base/limit information;
- Accounting Information
  - time limits, process number; owner
- I/O Status Information
  - list of I/O devices allocated to the process;

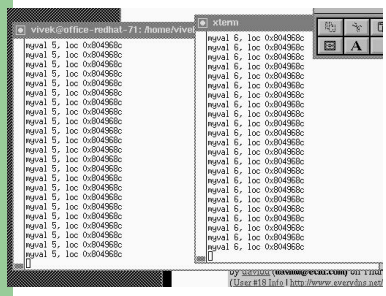
What Does This Program Do?

```
int myval;
int main(int argc, char *argv[])
{
   myval = atoi(argv[1]);
   while (1)
        printf("myval is %d, loc 0x%lx\n",
                myval, (long) &myval);
}
```

Now simultaneously start two instances of this program

- Myval 5
- Myval 6
- What will the outputs be?

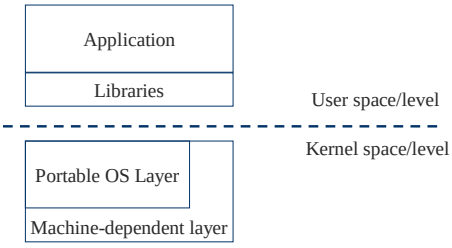
Here's The Output



Instances Of Programs

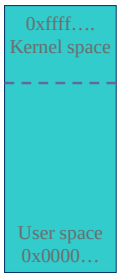
- The address was always the same
- The values were different
  - Implies that the programs aren't seeing each other
  - But they think they're using the same address
- Conclusion: addresses are not absolute
- Implication: memory mapping
- What's the benefit?

Remember This?



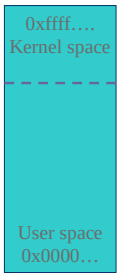
Address Space

- One (common) approach
  - Kernel is high memory
  - User is low memory
- What restrictions apply?
- read(f, buf, nbytes)



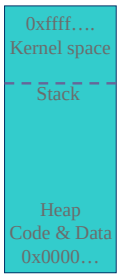
More Address Space

- Program segments
  - Text
  - Data
  - Stack
  - Heap
- What is stack?
- What is heap?
- What is data?
- What is text?



One Common Layout

- Lots of flexibility
  - Allows stack growth
  - Allows heap growth
  - No predetermined division



Process Control Block (PCB)

Process management	Memory management	File management
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Fields of a process table entry

Context Switch

- Switch CPU from one process to another
- Performed by scheduler (chapter 2.5)
- It includes:
  - save PCB state of the old process;
  - load PCB state of the new process;
  - Flush memory cache;
  - Change memory mapping (TLB);
- Context switch is expensive(1-1000 microseconds)
  - No useful work is done (pure overhead)
  - Can become a bottleneck
  - Real life analogy?
- Need hardware support

Summary

- What is process
- Process states and model
- What information in a process
- Process address space
- Context switch
- Next lecture: thread

Question for the next lecture

- What is the similarity and difference between process and thread?
- Reminders:
  - Test quiz
  - Nacho tutorial