Parallel Systems

(Soon to be basic OS knowledge)

Parallelism

✦ Multiple processes concurrently

Pseudo-Parallelism

Process 1 → CPU1 → CPU1 → CPU1

Process 2 → CPU1 → CPU1 → CPU1

True Parallelism

Process 1 → CPU1

Process 2 → CPU2

Parallel Hardware

Registers
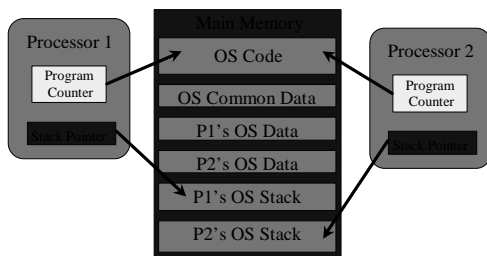CPU1

Registers
CPU2

Memory

Disk Controller — Disk
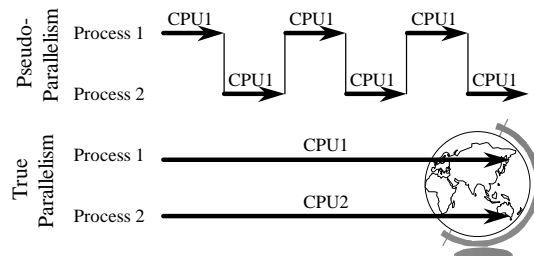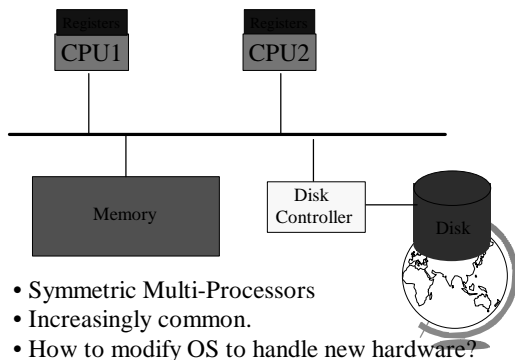
• Symmetric Multi-Processors
• Increasingly common.
• How to modify OS to handle new hardware?

Two Operating Systems

✦ Divide memory in two
✦ Run an independent OS in each
✦ Each has it's own processes
✦ Drawbacks
  – Twice as much memory used for OS
  – IPC tough
  – Who controls memory and disk? (convenient)
  – Inefficient scheduling (efficient)

Sharing the Operating System

Processor 1
Program Counter
Stack Pointer

Main Memory
OS Code
OS Common Data
P1's OS Data
P2's OS Data
P1's OS Stack
P2's OS Stack

Processor 2
Program Counter
Stack Pointer

Shared? (see sample code)
  stack   process table
  current process device queues

Race Conditions!

Example Multiprocessor OSes

✦ Almost all new OSes!
✦ Designed from start
  – Windows NT
  – Mach

✦ Unix
  – AT&T System V
  – Sun Solaris
  – HP Unix
  – Digital Unix
  – IBM AIX
  – SGI Irix

Threads

Software Multi-Processors

Threads (Lightweight Processes)

✦ Basic unit of CPU utilization
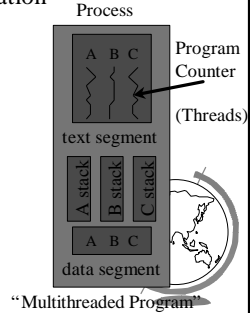- ("What?!" you say)

✦ Own
- program counter
- register set
- stack space

✦ Shares
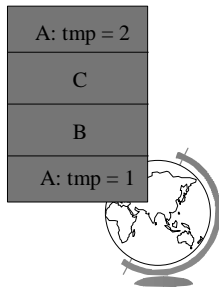- code section
- data section
- OS resources

Process

A B C

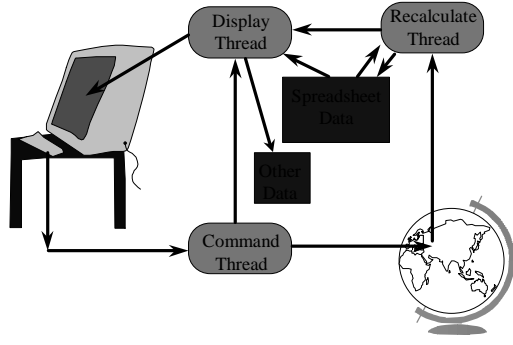Program Counter

(Threads)

text segment

A stack B stack C stack

A B C

data segment

"Multithreaded Program"

Stack

```
A(int tmp) {
  B();

  printf(tmp)
  ;
}
B() {
  C();
}
C() {
  A(2);
}
```

| A: tmp = 2 |
| C |
| B |
| A: tmp = 1 |

Example: A Threaded Spreadsheet

Display Thread

Recalculate Thread

Spreadsheet Data

Other Data

Command Thread

What Kinds of Programs to Thread?

✦ Independent tasks
- ex: debugger needs gui, program, perf monitor…
- especially when blocking for I/O!

✦ Single program, concurrent operation
- Servers
  ◆ ex: file server, web server
- OS kernels
  ◆ concurrent requests by multiple users -- no protection needed in kernel

---

Thread Benefits

✦ "What about just using processes with shared memory?"
- fine
- debugging tougher (more thread tools)
- processes slower
  ◆ 30 times slower to create on Solaris
  ◆ slower to destroy
  ◆ slower to context switch among
- processes eat up memory
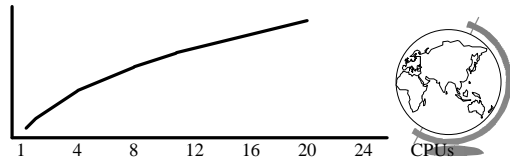 ◆ few thousand processes not ok
  ◆ few thousand threads ok

Threads Standards

✦ POSIX (Pthreads)
  – Common API
  – Almost **all** Unix's have thread library
✦ Win32 and OS/2
  – very different from POSIX, tough to port
  – commercial POSIX libraries for Win32
  – OS/2 has POSIX option
✦ Solaris
  – started before POSIX standard
  – likely to be like POSIX

Do they Work?

✦ Operating systems
  – Mach, Windows NT, Windows 95, Solaris, IRIX, AIX, OS/2, OSF/1
  – Millions of (unforgiving) users
✦ NFS, SPECfp, SPECint

| 1 | 4 | 8 | 12 | 16 | 20 | 24 | CPUs |

Levels of Threads

Process A   Process B

● User Level Thread
○ Thread
● Kernel Thread