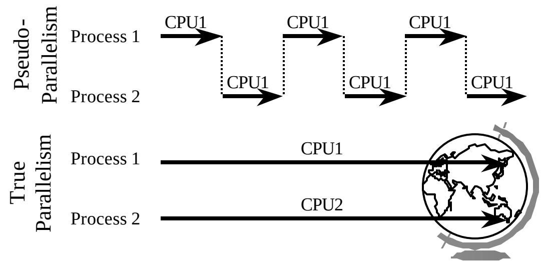
Operating Systems

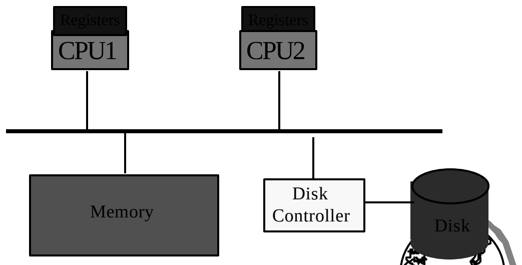
Parallel Systems
(pp. 515-521)

Parallelism

- Multiple processes concurrently



Parallel Hardware

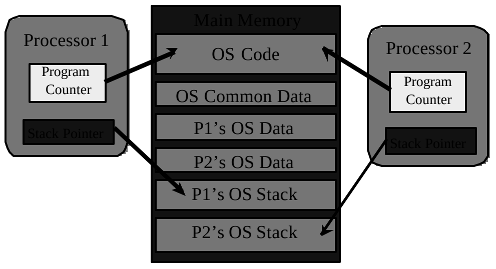


- Symmetric Multi-Processors
- Increasingly common.
- How to modify OS to handle new hardware?

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



Shared?
stack
current process
process table
device queues

Race Conditions!

SOS: Multi-Processor Support

- In StartUsingProcessTable()
  - What is the `exchangeword` mechanism similar too?
  - We busy wait. Is this ok? Why or why not?
- In FinishUsingProcessTable()
  - We don't protect setting the Flag. Is this ok? Why or why not?
- In SelectProcessTable()
  - Why do we have the variable `return_value`
- What other parts of the OS would need protection?

Example Multiprocessor OSes

- Almost all new OSes!
- Designed from start
  - Windows NT/2000
  - Mach
- Unix
  - AT&T System V
  - Sun Solaris
  - HP Unix
  - OSF Unix
  - IBM AIX
  - SGI Irix
  - Linux

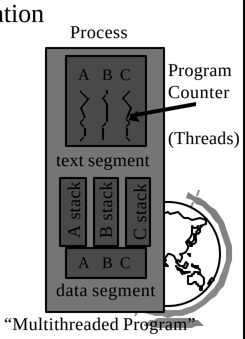
Threads

Software Multi-Processors

(Ch 2.2)

Threads (Lightweight Processes)

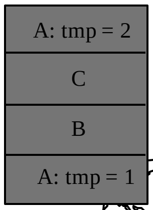
- Basic unit of CPU utilization
  - ("What?!" you say)
- Own
  - program counter
  - register set
  - stack space
- Shares
  - code section
  - data section
  - OS resources



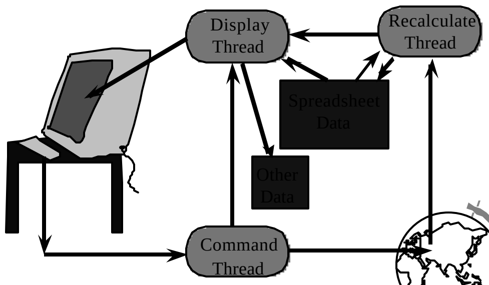
Stack

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

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



Example: A Threaded Spreadsheet



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 system requests by multiple users

Thread Benefits

- "What about just using multiple 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

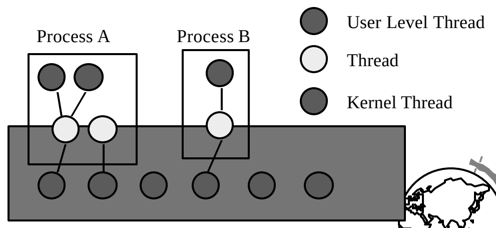
Threads Standards

- 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 same as POSIX

SOS: Thread Implementation

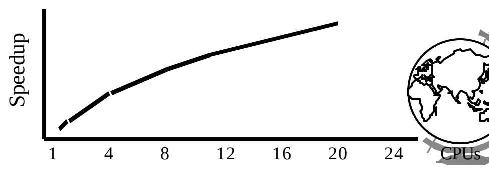
- Why doesn't the Process have a state anymore?
  - Does a process have to have threads?
- What new system calls might be useful for support of threads?
- What new scheduling criteria might the Dispatcher use when scheduling threads?

Levels of Threads



Do they Work?

- Operating systems
  - Mach, Windows NT, Windows 95, Solaris, IRIX, AIX, OS/2, OSF/1
  - Millions of (unforgiving) users
- NFS, SPEC


Speedup
CPUs