

# Introduction

**Real stuff:** Windows XP

Key operating system data structures are managed as objects.

Dispatcher objects:

- *process*. Encapsulates a virtual address space.
- *thread*. Associated with a process, scheduled by the kernel dispatcher.
- *event*. Record an event occurrence, signal waiting threads.

- *timer*. Signal timeouts (time limits or periodic activity).
- *mutex*. Provide dead lock free mutual exclusion.

## Course Organization

- Discuss concepts, compare existing and proposed solutions (2/3).
- Analyze a simple instructional OS (Nachos) in enormous detail to get a feel for implementations of the concepts in a real system.

## What is an OS?

*Two aspects:*

- A set of procedures that *coordinates* resources (CPU, memory, I/O devices, etc.) and enables a group of people to share a computer (computers) *efficiently*.
- A library of subroutines that provides a friendly *interface* between users and a computer (computers).

*Two functions:* coordinator, standard library.

The program, called the kernel, running at all times.

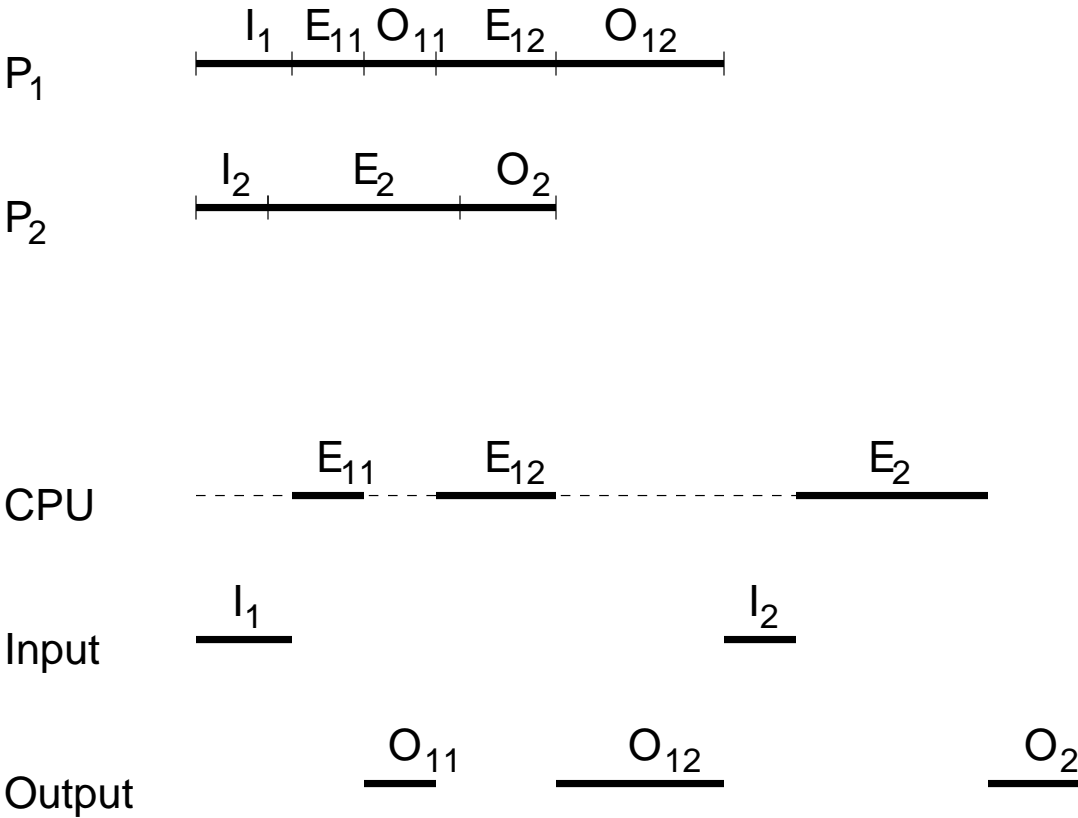
Most of this course will deal with the coordination aspect.

## Evolution

**Batch processing** Systems that execute programs serially with no direct interaction between the user and the computer.

- The OS always resides in memory.
- Single user, no execution and I/O overlap.
- OS = a program to load and run user jobs, take dumps, one job to the next.
- The CPU is often idle because I/O devices are much slower than the CPU.

Example

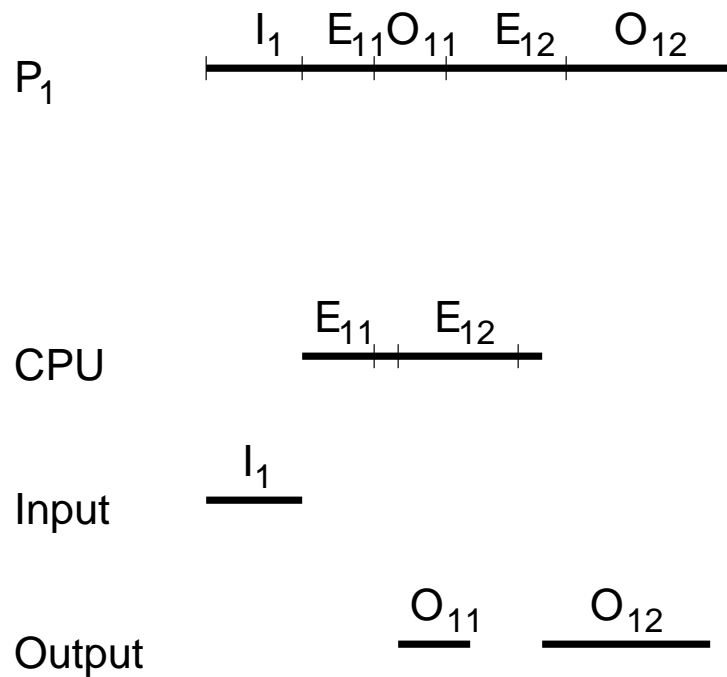


**Buffering** Overlap the I/O of one job with its own computation.

**I/O Interrupts** The interrupt handler signals the user process upon the completion of I/O.

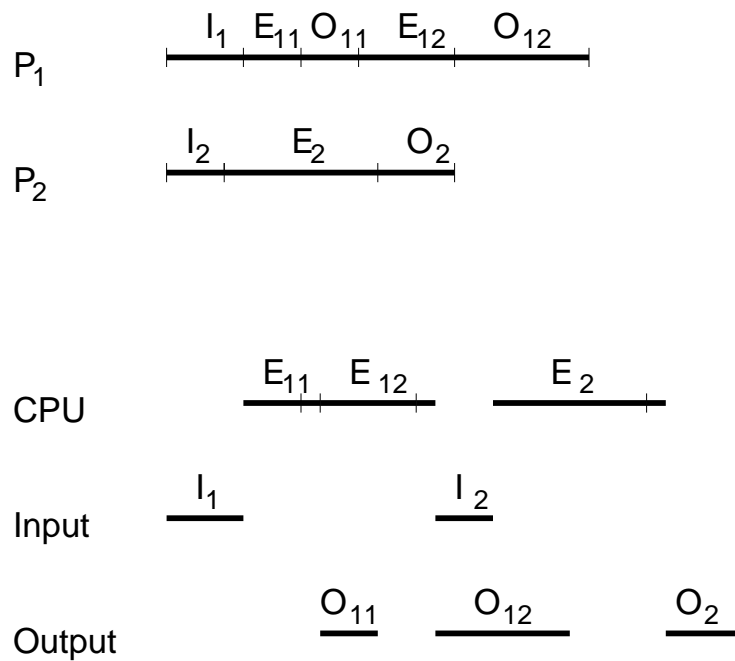
- Synchronous. The user process makes an I/O request and waits until the I/O completion. The user process must know the I/O latency, so it knows how long it should wait.
- Asynchronous. An I/O request returns without waiting for the I/O completion. An I/O interrupt is scheduled at the completion of I/O. The interrupt handler signals the user process when the I/O is done. This allows concurrent I/O operations to several devices.

Example.



**Direct Memory Access (DMA)** Reduce I/O interrupt frequency for high-speed devices by directly transferring a block of data (instead of one character or one word) from/to its buffer to/from memory.

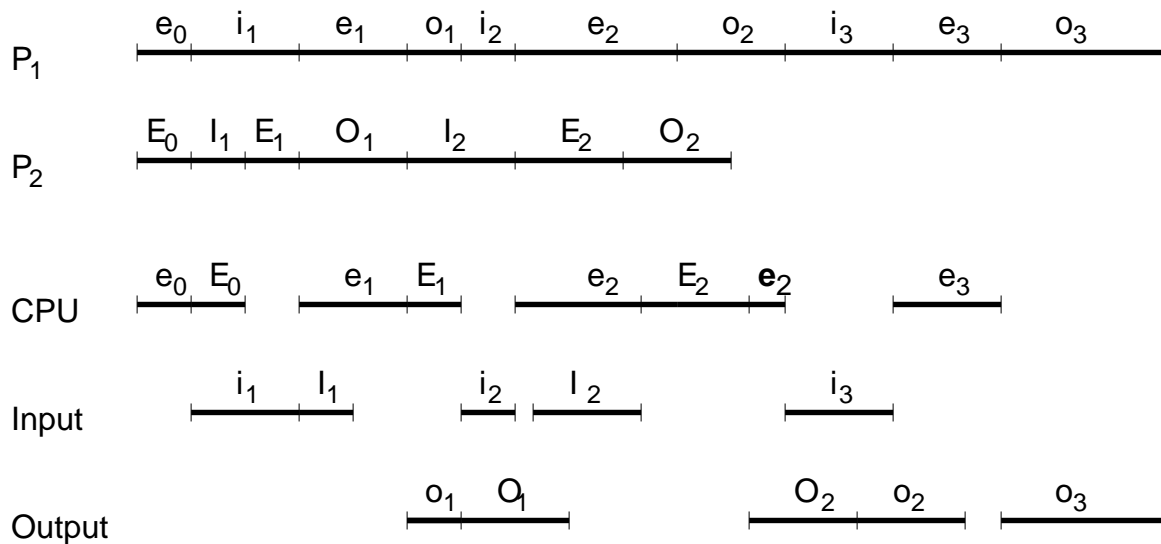
**Multiprogramming** Systems that are designed to concurrently execute more than one task.



Issues:

- Memory protection + relocation.
- OS began to be an important science.

**Timesharing (or Multitasking)** Systems that allow multiple users (programs) to run concurrently. The system switches from one user to another. Examples, MULTICS at MIT and UNIX at Bell Lab (1970). Example.



Issues:

- Response time.
- Thrashing.

**Multiprocessor Systems** High speed, better performance/price ratio, fault tolerant.

Symmetric multiprocessing (SMP). All processors run the same OS, no master-slave relationship, share memory.

Issues:

- data consistency.
- load balancing.
- I/O bottle-neck.

**Cache coherency** An update to a variable in one cache must be immediately reflected in all other caches that hold the variable.

**Distributed Systems** Two or more individual systems coupled together.

Client-server systems. Serves requests from clients.

Concurrency (timesharing)

The CPU switches between processes. Many users can run at the same time as if each had a computer, although the CPU executes one instruction at a time.

Parallelism (multiprocessor)

Multiple processors run one or more programs in parallel. Multiple instructions can be executed on CPUs at the same time.

**Real-Time Systems** Operations have time limits on the operation of a processor or the flow of data. For example, sensor control.

Hard real-time system. Hard time limit on critical tasks.

Soft real-time system. A critical task get priority over other tasks and retains that priority until it completes.

**Handheld Systems** Small size, e.g., personal digital assistants (PDAs). Small memory, slow processors.

## Components of Operating System

- Process management: OS allows several users to be working at the same time, as if each had a private personal computer. Or, one user can be doing many things at the same time.

Operations: create, delete, join, go-to-sleep

- Memory management: OS coordinates the uses of memory shared by several processes. It swaps information back and forth between disk and main memory so the system can run even the total memory needed by all processes is greater than the total size of main memory.

Operations: allocate, deallocate, map processes to the memory.

- Input output: OS keeps the CPU busy while an I/O device is working.  
Operations: I/O interrupts, buffering, device drivers.
- File system: OS coordinates the usage of space for files so that all files can fit on the same disk.  
Operations: create/delete files and directories, open/close files, read/write files and directories, map files and directories to disk.
- Network: OS allows groups of workstations to communicate and work together.  
Operations: Send and receive messages to and from network.

## Hardware Protection

- Dual-mode operation: User mode and kernel (monitor, supervisor) mode. Privileged instructions can only be executed in kernel mode.
- I/O protection: All I/O instructions are privileged. A user program issues I/O instructions through operating system.
- Memory protection: Separate user programs from operating system. Separate user programs from each other.
- CPU protection: A time slice, a maximum time that each process is allowed to run before the next process runs on the CPU.

## Characteristics of current operating systems

- Dinosaur: enormous (100k's of lines, 100-1000 man-years).
- Maze: complex (asynchronous, conflicting needs of different users, hardware peculiarity).
- Spaghetti: poorly understood (systems outlive their builders, many bugs, behavior is hard to predict, tuning is often done by guessing).

## Why are operating systems interesting?

- Combining many things such as languages, hardware, data structures, algorithms.
- Creating illusions like a wizard making computer appear to be more than it really is.
  - Memory as large as disk
  - Single processor running multiple programs concurrently

- Managing large, complex systems (including users and programs), often in conflict needs (interactive vs. batch, friendly vs. secure, small jobs vs. big jobs).
- Protecting users from each other; security issues pose tough problems especially in networking environment.
- Presenting general techniques. So you can learn and apply them elsewhere.