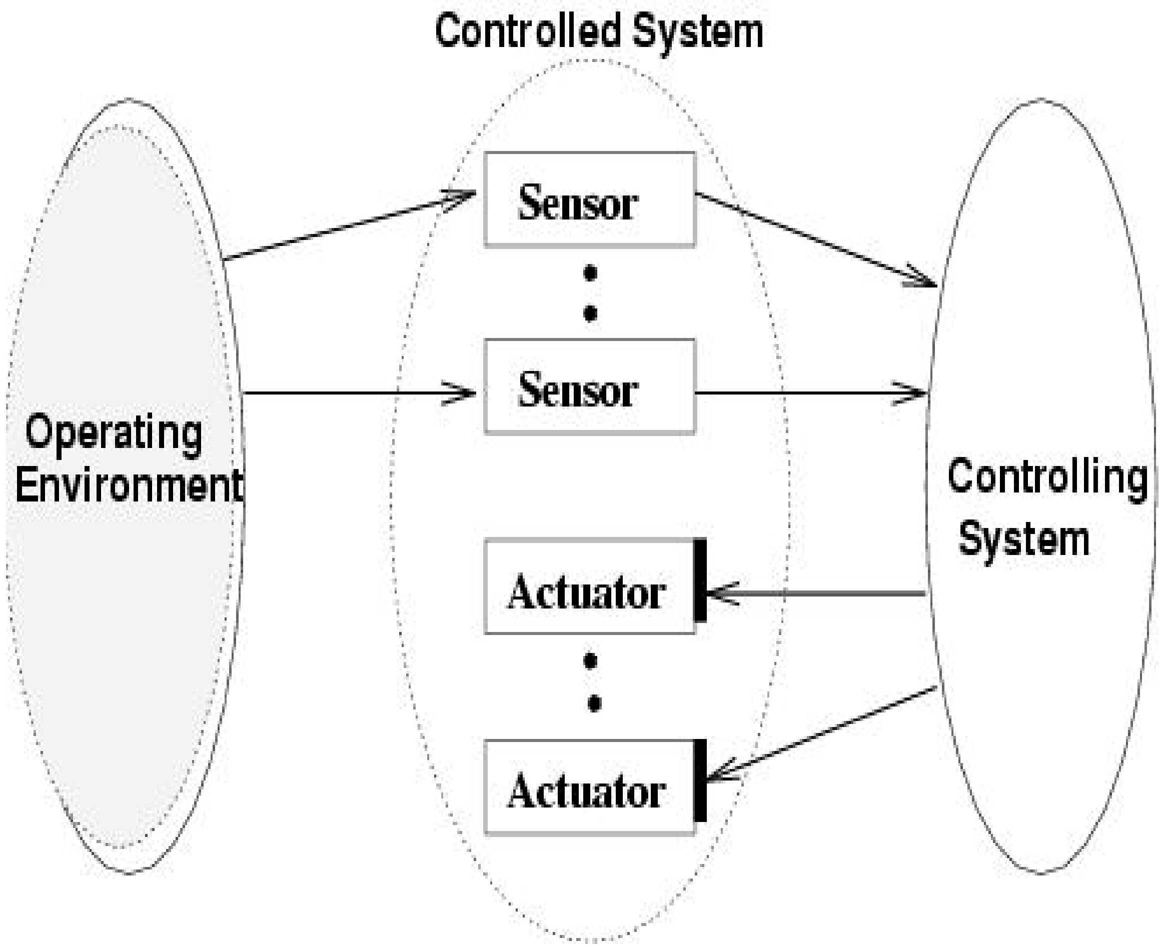
Real-Time Systems

Basic Concepts

Prof. Jiyao An

Typical RTS


Controlled System
Sensor
Sensor
Actuator
Actuator
Operating Environment
Controlling System

Example: Car

- ***Mission:*** Reaching the destination safely.
- **Controlled System:** Car.
- **Operating environment:** Road conditions and other cars.
- **Controlling System**
  - *Human driver:* Sensors - Eyes and Ears of the driver.
  - *Computer:* Sensors - Cameras, Infrared receiver, and Laser telemeter.
- **Controls:** Accelerator, Steering wheel, Break-pedal.
- **Actuators:** Wheels, Engines, and Brakes.

Definitions

- System: black box with n inputs and m outputs.
- Response time: time between presentation of a set of inputs and the appearance of the corresponding outputs.
- Events: Change of state causing a change of flow-of-control of a computer program.

Definitions

- synchronous: events occur at predictable times in the flow-of-control.
- asynchronous: unpredictable (interrupts!).
- state-based vs. event-based:
  - plane wing is at an angle of 32º (state)
  - plane wing moved up 4º (event)
- deterministic system: for each possible state and each set of inputs, a unique set of outputs and next state of the system can be determined.

More Definitions

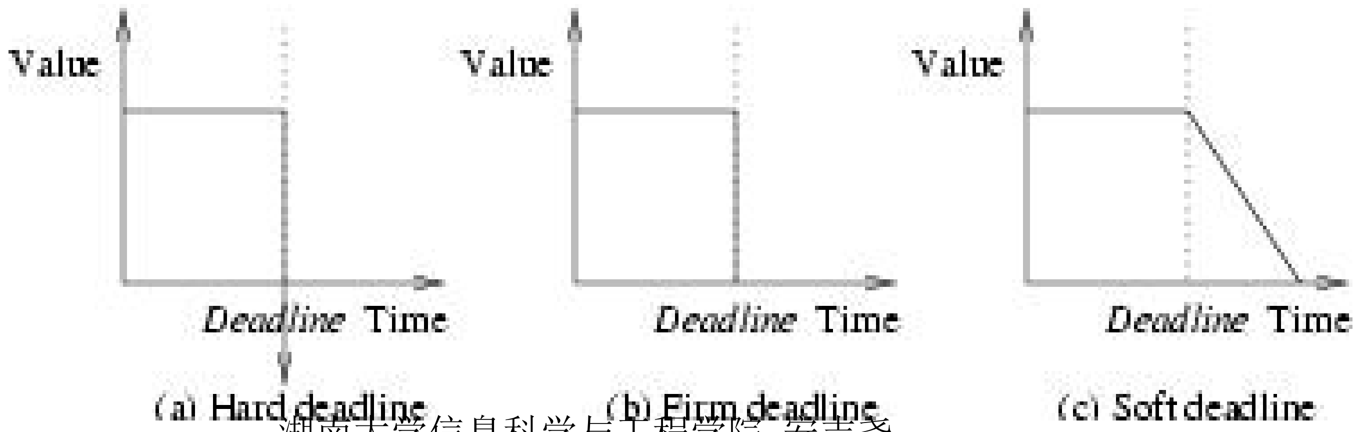
- Utilization: measure of ‘useful’ work a system performs.
- RTS: Correctness depends on results PLUS the time of delivery! Failure can have severe consequences.
- What are real-time systems? Planes, cars, washer, video player, thermostat, video games, weapons,...
- Related: QoS management, resource management, adaptive systems, embedded systems, pervasive and ubiquitous computing, ...

Other Definitions

- Oxford Dictionary of Computing: “Any system in which the time at which output is produced is significant. This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement- The lag from input time to output time must be sufficiently small for acceptable timeliness”.
- Burns and Wellings 2001: “Any information processing activity or system which has to respond to externally generated input stimuli within a finite and specified delay”.
- Laplante (1993): “A real-time system is a system that must satisfy explicit (bounded) response-time constraints or risk severe consequences, including failures”.

Hard versus Soft

- HARD: miss a deadline and you're in trouble! (planes, trains, factory control, nuclear facilities, ...)
- SOFT: try to meet deadlines, but if not, system still works, although with degraded performance (multimedia, thermostat, ...)
- FIRM: late results are worthless, but you are not in trouble



Other categorization

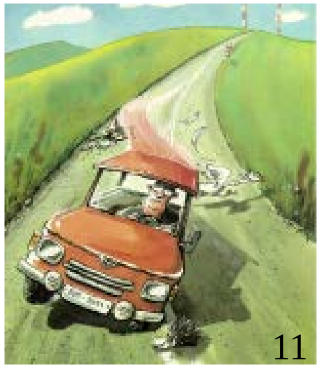
- Degrees of real-time (subjective!):
  - slightly: payroll systems (generate checks)
  - a little more: disk driver software
  - considerably more: credit-card authorizations, ATM withdrawals, airline booking
  - highly: combat system, stability control in airplanes, ABS（Anti-locked Braking System）

More Definitions

- Reactive: system ‘reacts’ to environmental changes (temperature changes).
- Embedded: specialized hardware and software, because GP-systems lack real-time capabilities.

Example: cruise control

- Regulates speed of car by adjusting the throttle: driver sets a speed and car maintains it.
- Measures speed through device connected to drive shaft.
- Hard real-time: drive shaft revolution events.
- Soft real-time: driver inputs, throttle adjustments.



More examples

- cars: engine control, ABS, drive-by-wire
- planes: stability, jet engine, fly-by-wire
- computers: peripherals, applications
- military: weapons, satellites
- domestic: microwave, thermostat, dishwasher
- medical: pacemaker, medical monitoring
- protection: intruder alarm, smoke/gas detection

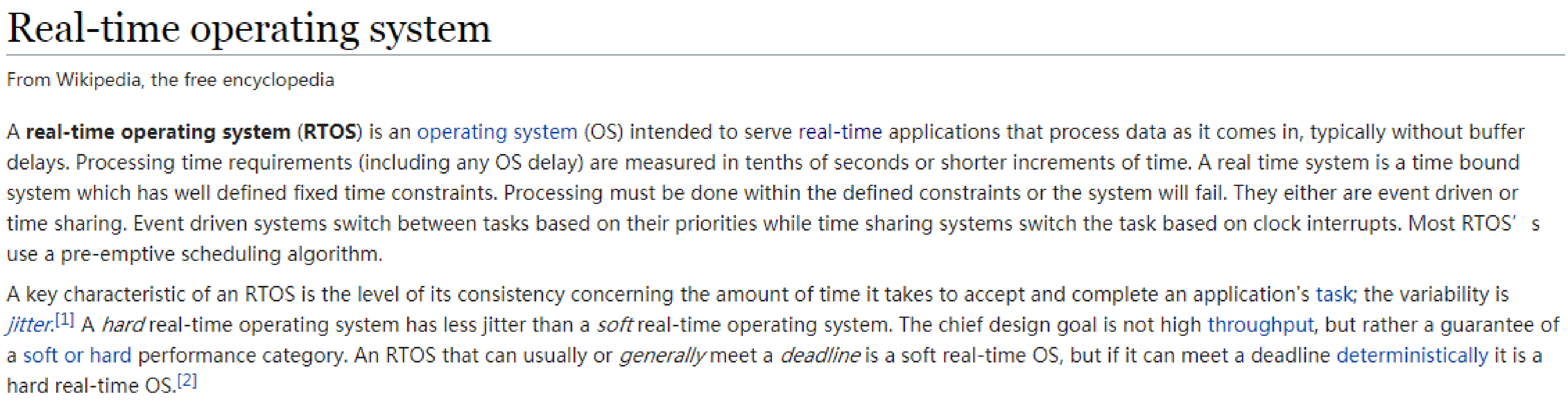
Characteristics of RT Systems

- size: small assembler code or large C++, Ada, ... code (example: 20 million lines of Ada for Intl. Space Station).
- concurrent control of separate components (model this parallelism with parallelism in your program).
- use of special purpose hardware and tools to program devices for this hardware in a reliable manner.

Common Misconceptions

- “real fast” is real-time: a computer system may satisfy an application’s requirement, but no predictability (no real-time resource management).
- hardware over-capacity is enough: again, without real-time resource management no appropriate balance of resource distribution.

real-time OS - wikipedia



Real-time operating system

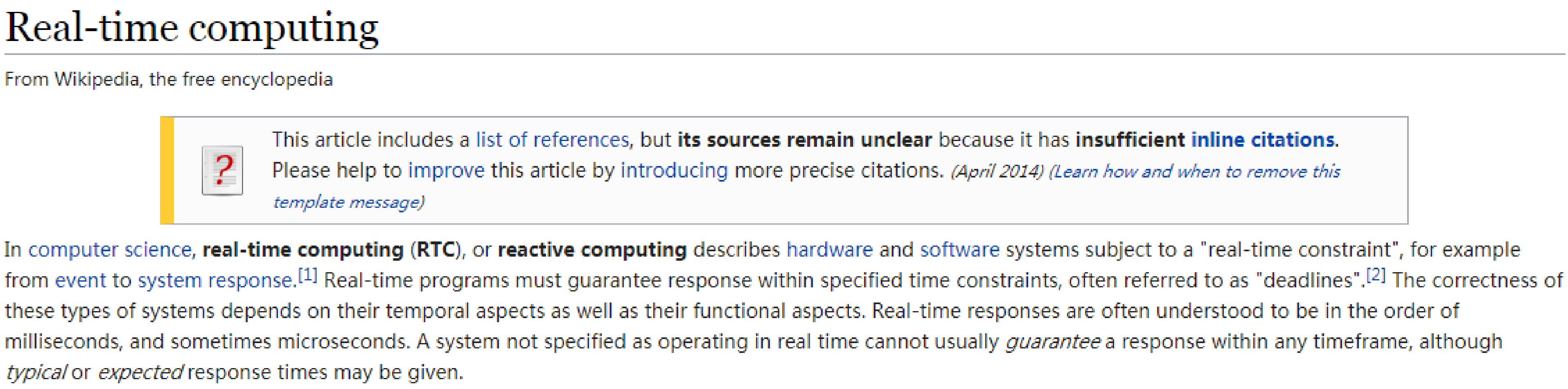
From Wikipedia, the free encyclopedia

A **real-time operating system** (**RTOS**) is an operating system (OS) intended to serve real-time applications that process data as it comes in, typically without buffer delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter increments of time. A real time system is a time bound system which has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail. They either are event driven or time sharing. Event driven systems switch between tasks based on their priorities while time sharing systems switch the task based on clock interrupts. Most RTOS' s use a pre-emptive scheduling algorithm.

A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task; the variability is *jitter*.[1] A *hard* real-time operating system has less jitter than a *soft* real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category. An RTOS that can usually or *generally* meet a *deadline* is a soft real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS.[2]

https://en.wikipedia.org/wiki/Real-time_operating_system

real-time computing - wikipedia



Real-time computing

From Wikipedia, the free encyclopedia

This article includes a list of references, but **its sources remain unclear** because it has **insufficient inline citations.** Please help to improve this article by introducing more precise citations. *(April 2014) (Learn how and when to remove this template message)*

In computer science, **real-time computing** (**RTC**), or **reactive computing** describes hardware and software systems subject to a "real-time constraint", for example from event to system response.[1] Real-time programs must guarantee response within specified time constraints, often referred to as "deadlines".[2] The correctness of these types of systems depends on their temporal aspects as well as their functional aspects. Real-time responses are often understood to be in the order of milliseconds, and sometimes microseconds. A system not specified as operating in real time cannot usually *guarantee* a response within any timeframe, although *typical* or *expected* response times may be given.

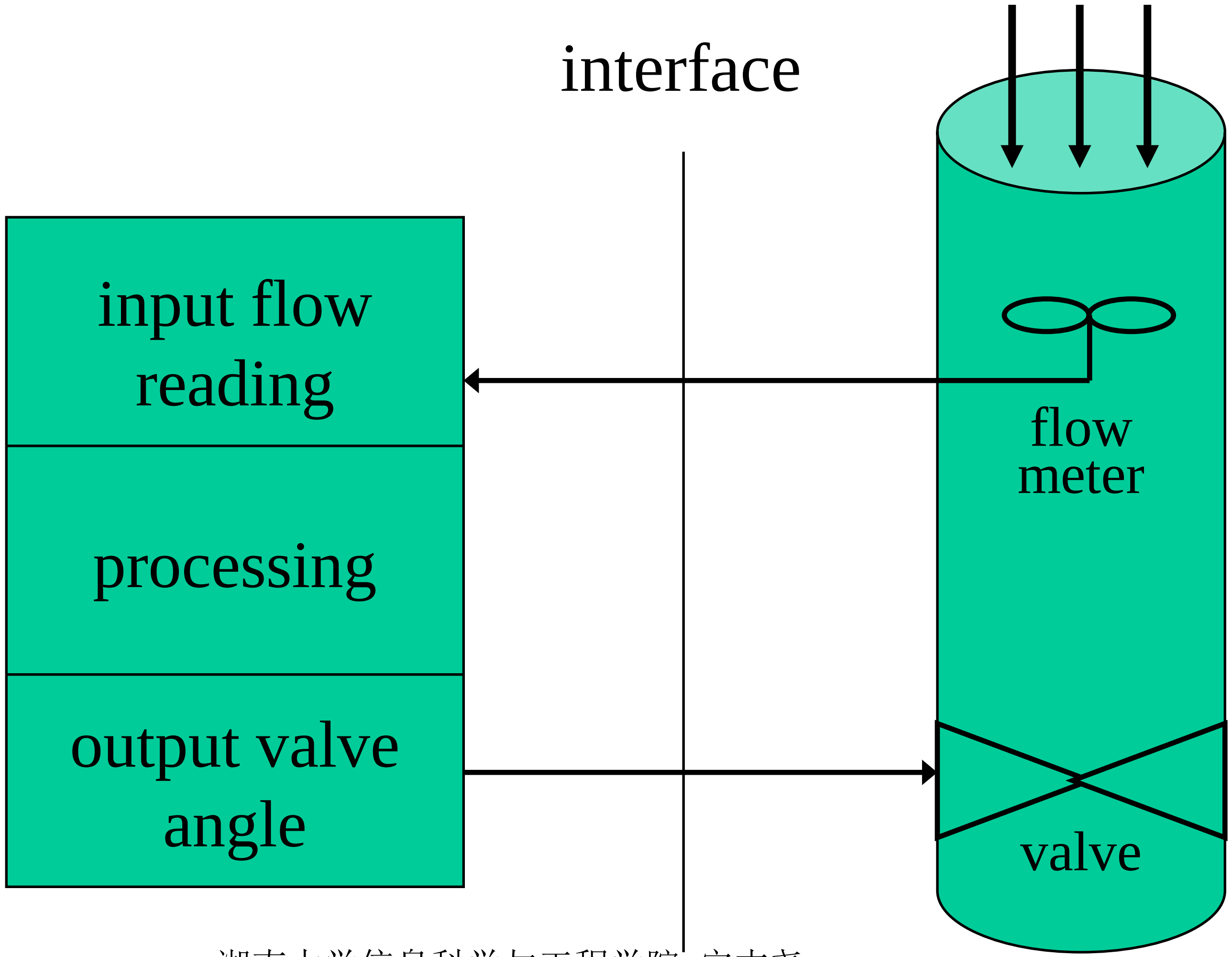
https://en.wikipedia.org/wiki/Real-time_computing

Predictability

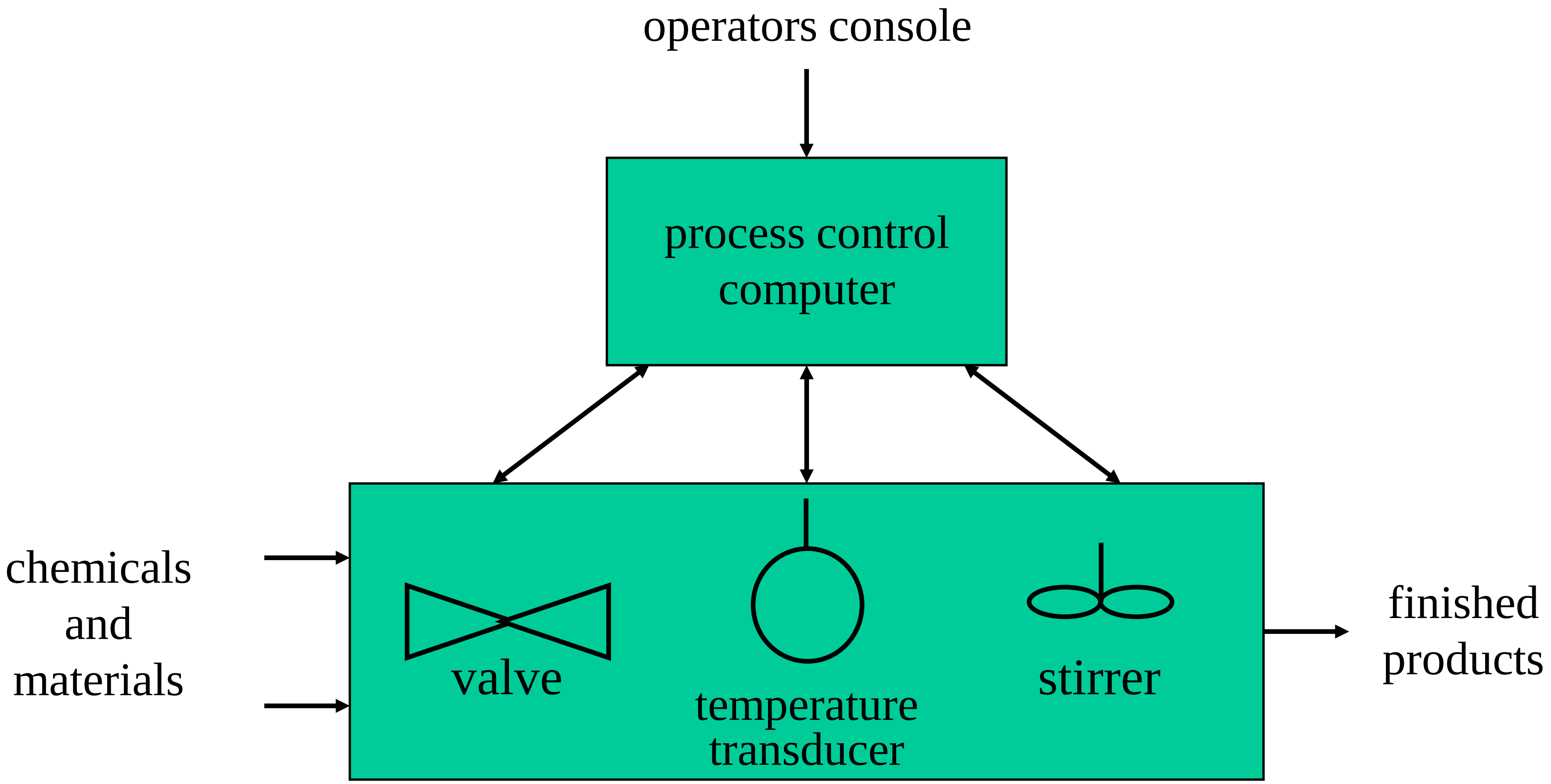
- The most common denominator that is expected from a real-time system is ***predictability***.
  - **The behavior of the real-time system must be predictable which means that with certain assumptions about workload and failures, it should be possible to show at "design time" that all the timing constraints of the application will be met.**

- For static systems, 100% guarantees can be given at design time.
- For dynamic systems, 100% guarantee cannot be given since the characteristics of tasks are not known a priori.
- In dynamic systems, predictability means that once a task is admitted into the system, its guarantee should never be violated as long as the assumptions under which the task was admitted hold.

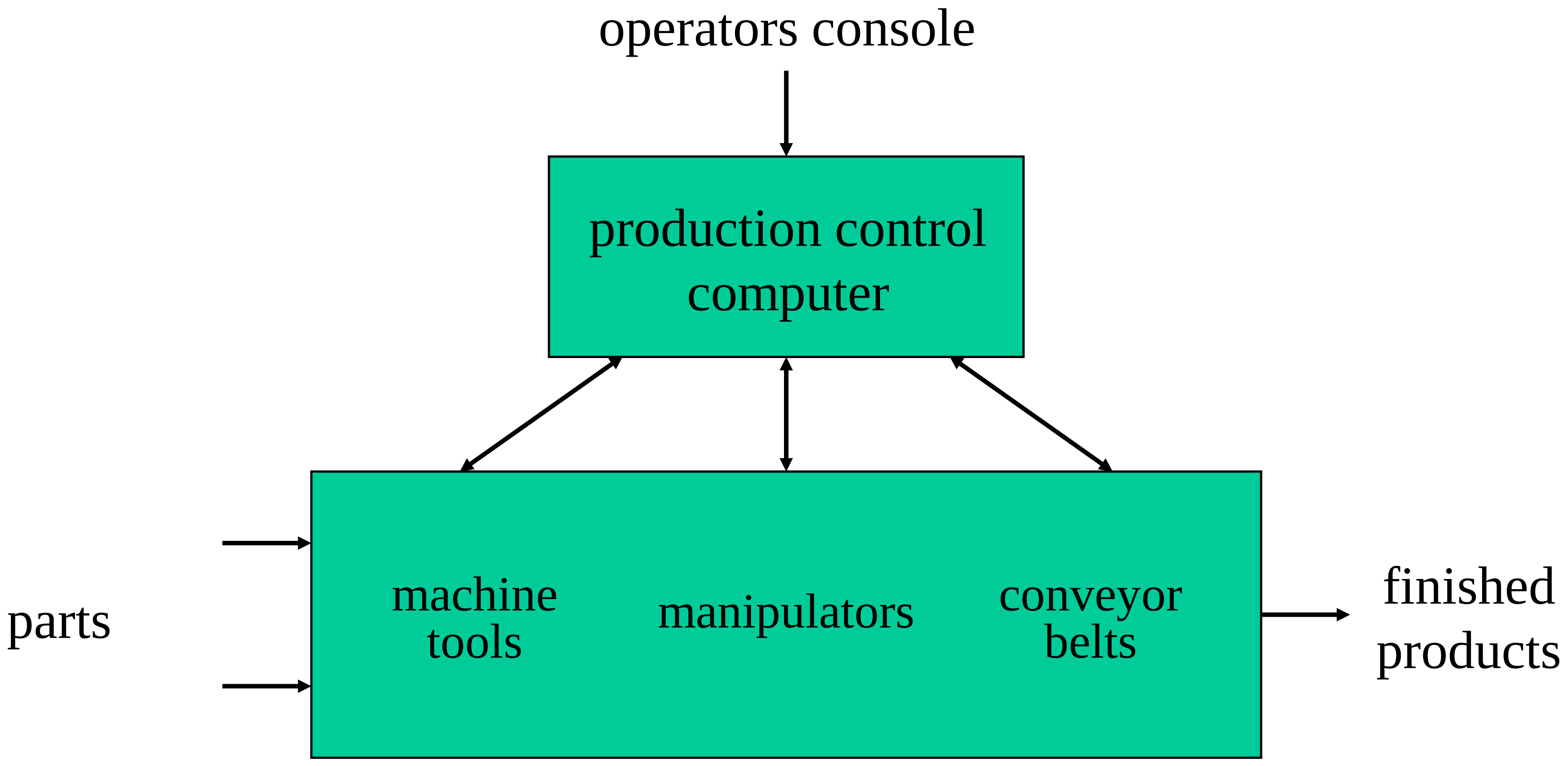
Simple Valve Control



Process Control

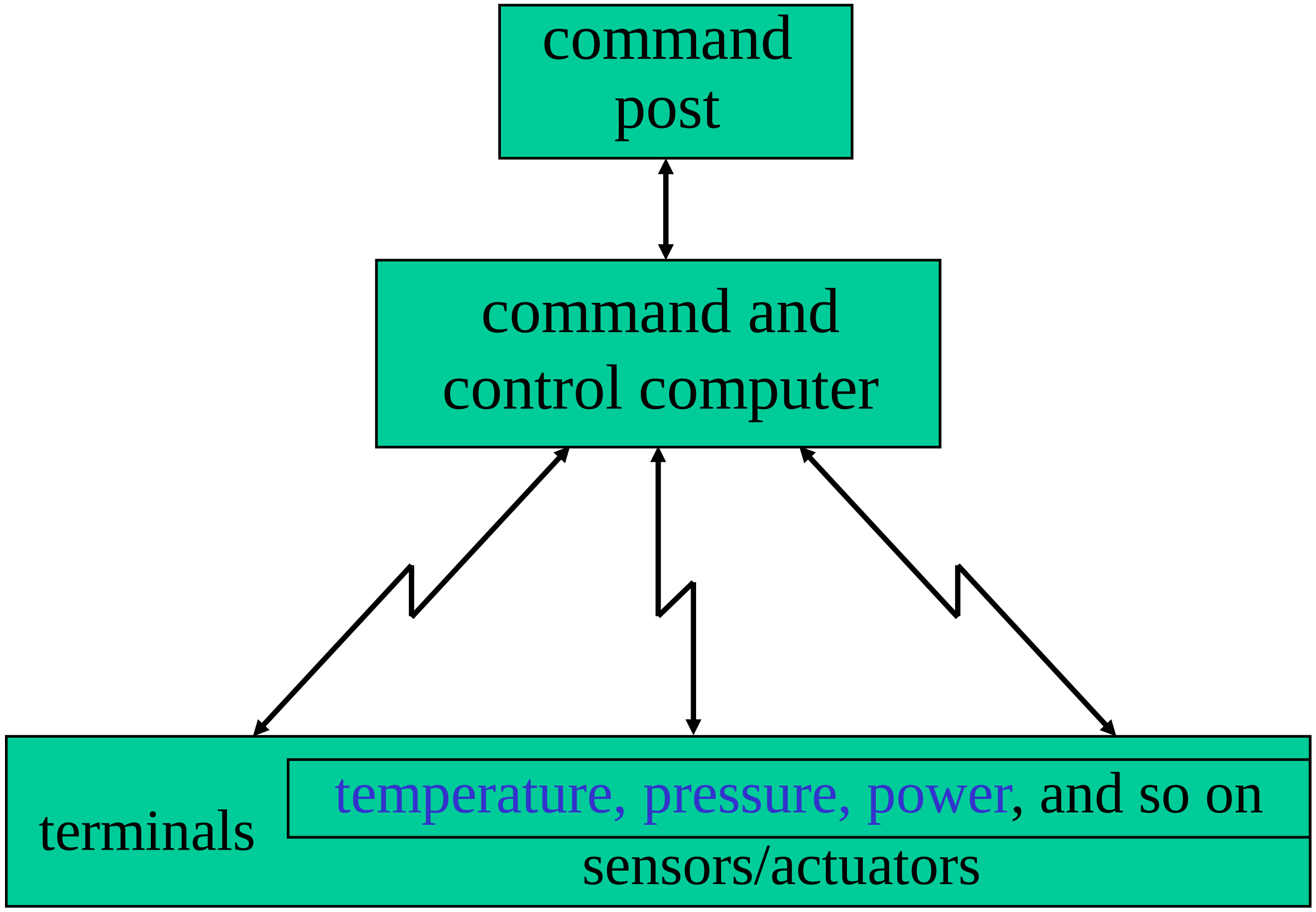


Manufacturing



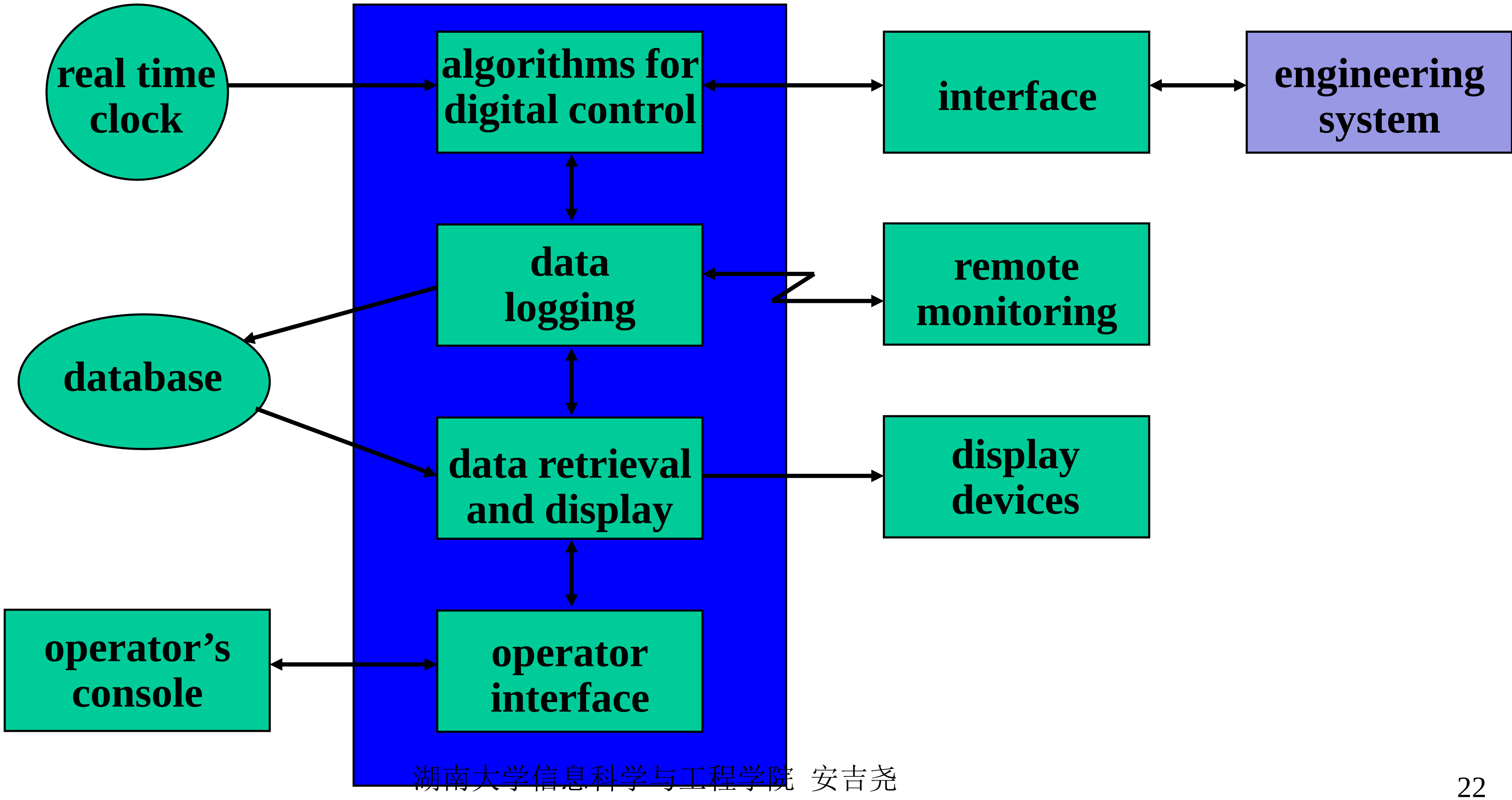
a production control system

CCC

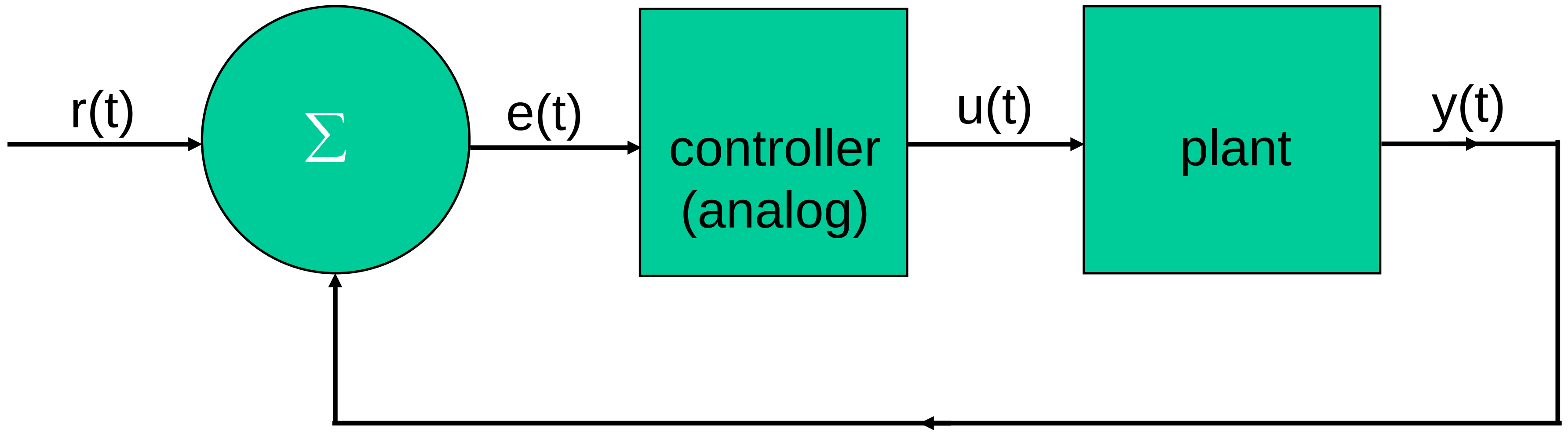

command post
command and control computer
terminals
temperature, pressure, power, and so on
sensors/actuators

a command and control system

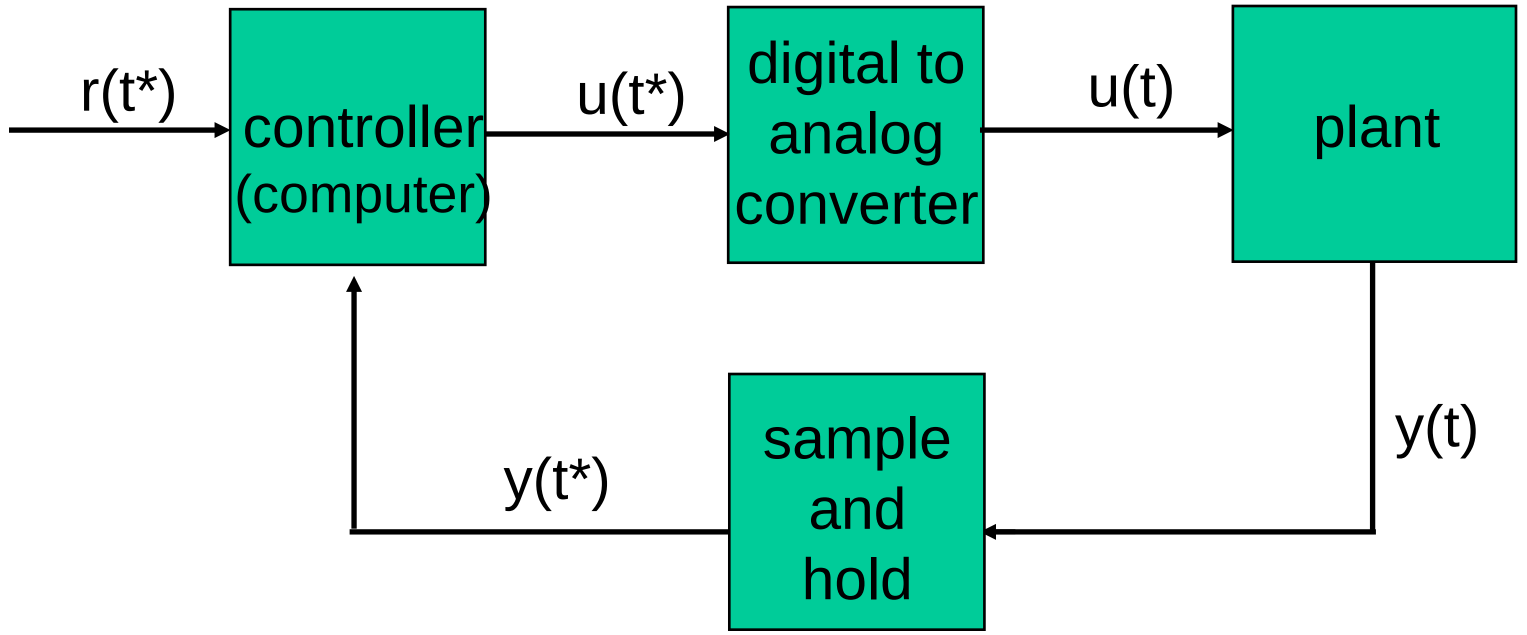
Industrial Embedded System



Feedback Control System

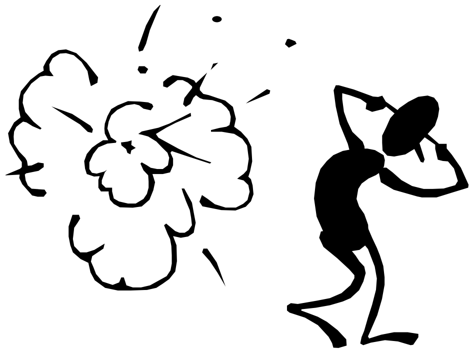


Digital Feedback Control



And More Definitions: Safety

- Safety (for environment and humans)
  - Burns & Wellings
    "Safety is the probability that conditions that can lead to mishaps do not occur whether or not the intended function is performed".



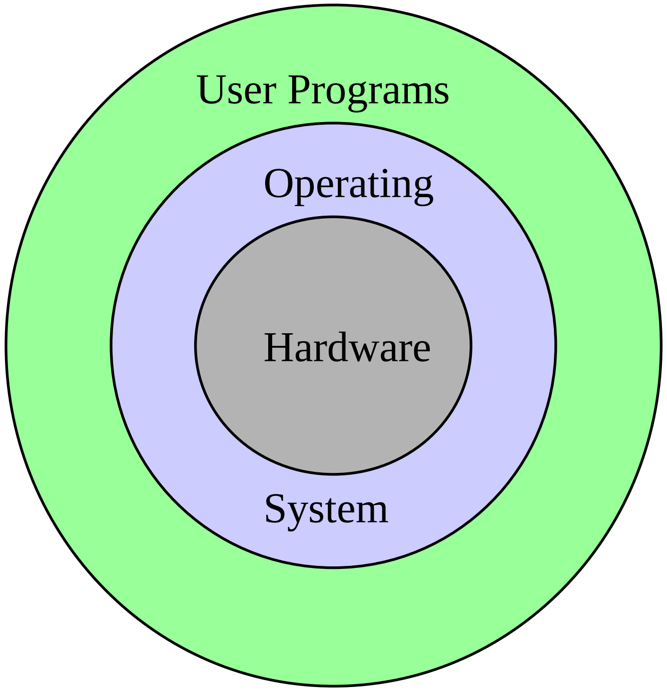
Reliability

- Reliability
  - Randell et al (1978)
    “a measure of the success with which the system conforms to some authoritative specification of its behavior“
- Safety and reliability often used interchangeably.
- Safety and reliability usually expressed in probabilities.
- Other frequently used term: dependability.

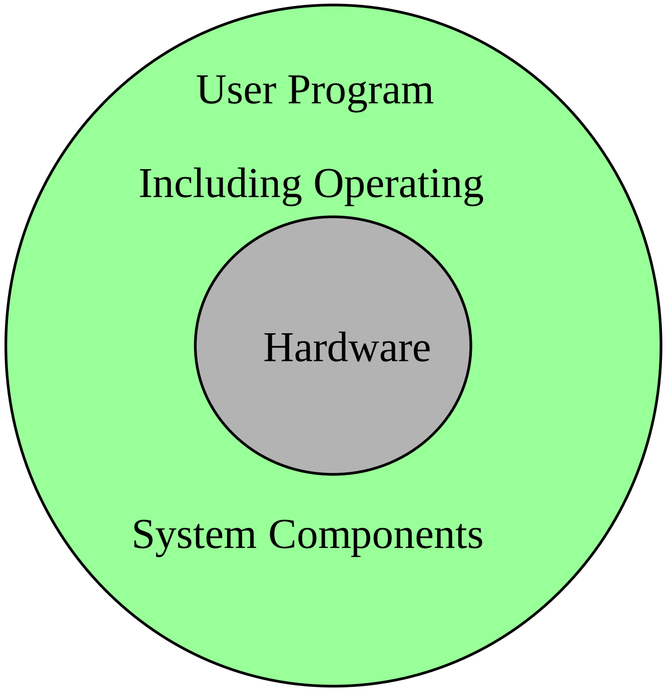
Example: Airplane

- The only SAFE airplane is one that never takes off!
- But it is not very reliable!

Operating Systems



Typical OS Configuration



Typical Embedded Configuration

Real-Time OSs

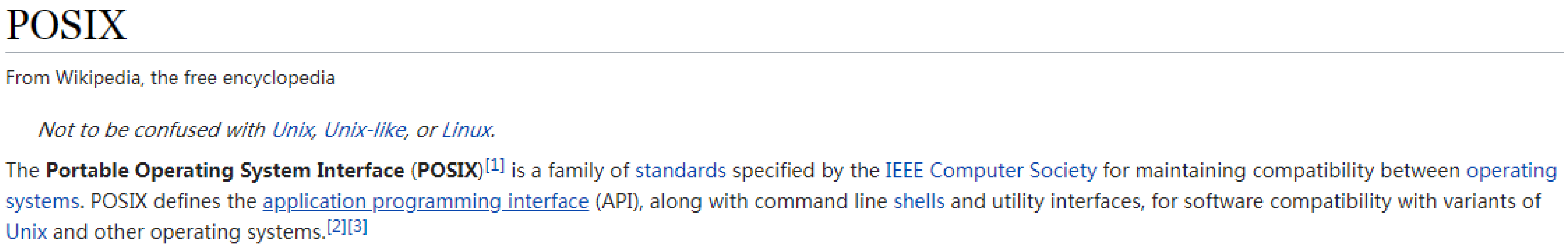
- Real-Time OS: VxWorks, QNX, LynxOS, eCos, DeltaOS, PSX, embOS, ...
- GPOS: no support for real-time applications, focus on ‘fairness’.
- BUT, people love GPOSs, e.g., Linux:
  - RTLinux (FSMLabs)
  - KURT (Kansas U.)
  - Linux/RT (TimeSys)

RT OSs

- Why?
  - Determinism / Predictability
    - Ability to meet **deadlines**
    - Traditional operating systems non-deterministic
- Standards?
  - Real-Time POSIX 1003.1
    - Pre-emptive fixed-priority scheduling
    - Synchronization methods
    - Task scheduling options

RT OSs

- Standards?
  - Real-Time POSIX 1003.1
    - Pre-emptive fixed-priority scheduling
    - Synchronization methods
    - Task scheduling options



POSIX

From Wikipedia, the free encyclopedia

Not to be confused with Unix, Unix-like, or Linux.

The **Portable Operating System Interface** (**POSIX**)[1] is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.[2][3]

https://en.wikipedia.org/wiki/POSIX

Examples

- Lynx OS
  - Microkernel Architecture
  - Provides scheduling, interrupt, and synchronization support
  - Real-Time POSIX support
  - Easy transition from Linux

Examples

- QNX Neutrino
  - Microkernel Architecture
    - Add / remove services without reboots
  - Primary method of communication is message passing between threads
  - Every process runs in its own protected address space
    - Protection of system against software failure
    - “Self-healing” ?

Examples

- VxWorks
  - Monolithic Kernel
    - Reduced run-time overhead, but increased kernel size compared to Microkernel designs
  - Supports Real-Time POSIX standards
  - Common in industry
    - Mars missions
    - Honda ASIMO robot
    - Switches
    - MRI scanners
    - Car engine control systems

Examples

- MARS (Maintainable Real-Time System)
  - Time driven
    - No interrupts other than clock
  - Support for fault-tolerant, redundant components
  - Static scheduling of hard real-time tasks at predetermined times
    - Offline scheduling
  - Primarily a research tool

Examples

- RTLinux
  - “Workaround” on top of a generic O/S
    - Generic O/S – optimizes average case scenario
    - RTOS – need to consider WORST CASE scenarios to ensure deadlines are met
  - Dual-kernel approach
    - Makes Linux a low-priority pre-emptable thread running on a separate RTLinux kernel
    - Tradeoff between determinism of pure real-time O/S and flexibility of conventional O/S
  - Periodic tasks only

Example: Interrupts

- Interrupt handling.
- Concurrent interrupts: queuing? priorities?
- Preemptive interrupts; enabling and disabling of interrupts.

Example: Concurrency

- Scheduling: priorities, time driven, event driven, task scheduling (RMS).
- Processes, threads.
- Synchronization: test-and-set instructions, semaphores, deadlocks (circular waits), ...

Example: Scheduling

- static: all scheduling decisions are determined before execution.
- dynamic: run-time decisions are used.
- periodic: processes that repeatedly execute
- aperiodic: processes that are triggered by asynchronous events from the physical world.
- sporadic: aperiodic processes w/ known minimum inter-arrival jitter between any two aperiodic events.

Preemptive vs. Non-preemptive

- **Preemptive Scheduling**
  - Task execution is preempted and resumed later.
  - Preemption takes place to execute a higher priority task.
  - Offers higher schedulability.
  - Involves higher scheduling overhead due to context switching.

- **Non-preemptive Scheduling**
  - Once a task is started executing, it completes its execution.
  - Offers lower schedulability.
  - Has less scheduling overhead because of less context switching.

Rate Monotonic Priority Assignment (RMA)

- each process has a unique priority based on its period; the shorter the period, the higher the priority.
- RMA proven optimal in the sense that if any process set can be scheduled (using preemptive priority-based scheduling) with a fixed priority-based assignment scheme, then RMA can also schedule the process set.

RMA

- Each task has a period T and run-time C.
- System utilization $U=\Sigma(C_i/T_i)$. Measure for computational load on the CPU due to the task set.
- There exists a maximum value of U, below which a task set is schedulable and above which it is not schedulable.
- RMA: Liu and Layland (1973):

$$\Sigma(C_i/T_i) <= n(2^{1/n}-1)$$

Real-Time Languages

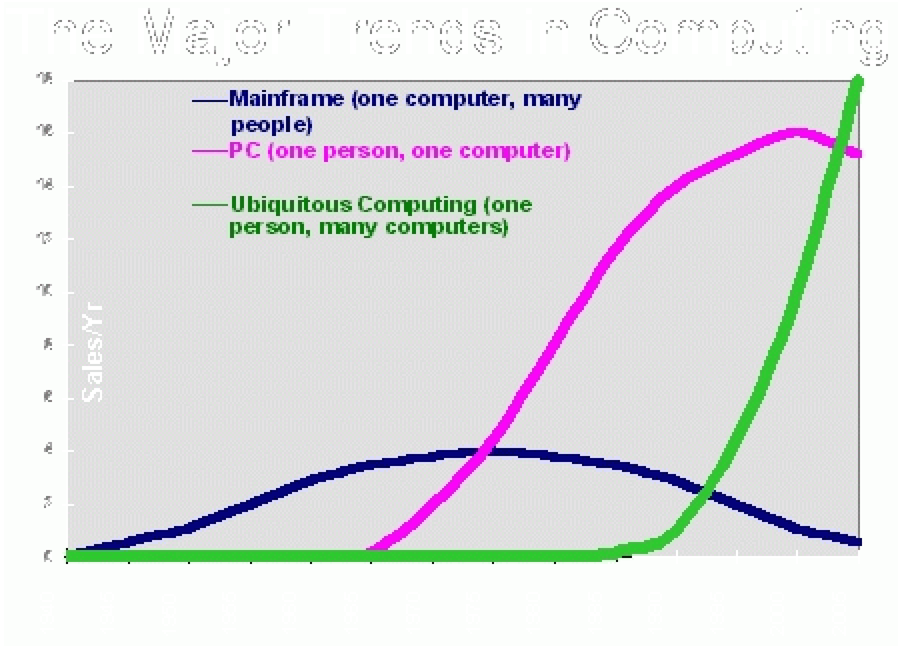
- Support for the management of time
  - Language constructs for expressing timing constraint, keeping track of resource utilization.
- Schedulability analysis
  - Aid compile-time schedulability check.
- Reusable real-time software modules
  - Object-oriented methodology.
- Support for distributed programming and fault-tolerance

Real-Time Databases

- Most conventional database systems are disk-based.
- They use transaction logging and two-phase locking protocols to ensure transaction *atomicity* and *serializability*.
- These characteristics preserve data integrity, but they also result in relatively slow and unpredictable response times.
- In a real-time database system, important issues include:
  - transaction scheduling to meet deadlines.
  - explicit semantics for specifying timing and other constraints.
  - checking the database system's ability of meeting transaction deadlines during application initialization.

What's Happening?

- Ubiquitous Computing: make computers invisible, so embedded , so fitting, so natural, that we use it without even thinking about it.



What’s Happening?

- Autonomous Computing:
  - self-configurable
  - self-adapting
  - optimizing
  - self-healing
- Building real-time systems:
  - toolkits, validation tools, program composition
  - Boeing 777: $4Billion, >50% system integration & validation!

What’s Happening?

- Soft real-time applications:
  - mainstream applications
  - notion of QoS
- Multi-dimensional requirements:
  - real-time, power, size, cost, security, fault tolerance
  - conflicting resource requirements and system architecture
- Unpredictable environments:
  - Internet (servers), real-time databases, ...

What’s Happening?

- Large-scale distributed mobile devices
  - connectivity
  - service location
  - scarce resources, resource sharing
  - power limitations

- Questions/Issues?