Operating systems

ELTE IK.

Dr. Illés Zoltán

zoltan.illes@elte.hu

Financed from the financial support ELTE won from the Higher Education Restructuring Fund of the Hungarian Government

We have talked earlier…

- Evolution of operating systems
- Notions of Operating system, structures
- Files,directories, filesystems
- Processes
  - process communications
  - critical sections, , semaphores
- Classical IPC problems
- Scheduling

What comes today…

- Controlling input/output devices
  - Type of I/O devices
  - Access of devices
    - Interrupt
    - DMA
    - I/O ports
- Base concepts of I/O programs
- Problems of allocating resources
  - Dead-locks
  - Base concepts
  - Recognize, prevent, avoid

Input-Output devices

- Block-type devices
  - Store information in a block with given size.
  - Block size is between 512 byte- 32768 byte.
  - They can be written or read independently from each other.
  - Addressable by blocks.
  - Such devices are: HDD, CD, magnetic tape device,etc.
- Character-type devices
  - Not addressable, characters (bytes) come and go one after the other
- Timer: exception, not block-type and not character-type

Model of I/O devices

- Though there are always some exceptions (e.g. timer), the device independent software-model of the operating system is based on this block-type, character-type model.
  - E.g. The filesystem is handling abstract block-type devices.
  - The magnetic tape is a block-type one, at the commend to read the N-th block it rewinds back and then forward.
  - The device dependent parts are the device drivers. (DDK)

The speed of I/O devices

- Keyboard: 10 byte/sec
- Mouse: 100 byte/sec
- 56k modem: 7kbyte/sec
- Scanner: 400kbyte/sec
- 52xCD ROM: 8 MB/sec (1xCD, 150kb/sec)
- Firewire : 50 MB/sec
- USB2: 60 MB/sec
  - USB3 500 MB/sec (theoritical, 4.8GBPS)
- SATA: 200 MB/sec
- SCSI UW4: 320 MB/sec
- PCI bus: 528 MB/sec

Device controllers

- To connect the I/O device to the computer (system bus) we use device controllers or adapters.
  - Video controller
  - Serial-parallel controller
  - USB controller
  - HDD controller
    - IDE, SATA, SCSI
- In the case of mainframes I/O is implemented with special I/O machines.
- CPU-device controller communication
  - Memory-mapped I/O, interrupt, DMA

I/O gate,memory-mapped I/O

- CPU uses two different ways of data exchange towards the outside world
  - To read, write I/O gates
    - IN register, port; To read the port data into the register (8 or 16 bits long number)
    - OUT port, register ; To write the register to the port
  - The registers, the data area of the I/O device is inside the memory.
- Maybe there are such environments where
  - Only I/O gates are used. (IBM 360)
  - Only in memories are I/O gates (PDP-11)
    - Memórialeképezésű I/O
  - Mixed (Intel x86, Pentium)

Interrupts I.

- Interrupt – the notion itself is connected to I/O devices!
  - If an I/O devices is ready for data communication it is indicated with an interrupt signal!
- Software or Hardware interrupt
  - Software: a machine-code instruction (int x) is executed, in the case of multitask it is called „trap”!
- There is a number of the interrupt!
  - Interrupt vectors (real mode, from address 0)
  - Interrupt handling routine.
  - INTR, NMI

Interrupt handling

- Usually devices have a state-bit stating that the data is ready.
  - You may observe it, but it is not the perfect solution.
  - It is a busy-waiting, not efficient, rarely used.
- The process of interrupt handling (IRQ)
  - The HW device indicates an interrupt request (INTR)
  - The CPU before the execution of next instruction interrupts its activity! (Precise, imprecise)
  - To execute the routine given by an ordinal number.
    - To read the required data and to execute the tightly connected work.
  - To return to the state before the interrupt request.

Priority of interrupts

- INTR – maskable, non maskable NMI
  - INTR priority of interrupts
    - The same or lower priority interrupts have to wait if they are arrived during the actual interrupt process!
  - NMI – only one is served: the greatest priority one
- In PC world 15 different interrupts
  - 2x8 channel controller
  - Manual adapter IRQ setting earlier, with jumper, with sw.
  - BIOS automatic interrupt assigning (Plug&Play)

Direct memory access (DMA)

- Direct Memory Access
  - Contains: a memory address register, a register to indicate the direction of delivery, others to decide the amount of data and to control
  - They can be accessed through standard in,out ports.
- Typical steps of working:
  1. CPU sets the DMA controller. (Registers)
  2. DMA asks the disk controller to execute the given operation.
  3. After the disk controller reads the data from its buffer the data is read/written to/from the memory through the system-bus..
  4. The disk-controller acknowledge the execution of the request.
  5. DMA indicates with an interrupt that the operation is finished.

I/O software goals I.

- Device independence
  - To be able to read e.g. a file from a HDD, CD, etc. with the same code, command.
  - Standard name usage, the name is a parameter (seales) the actual (real) device.
  - Logical mount
    - Unix: Floppy mounted to /home/fdd directory
    - Windows: Floppy mounted to a: name
- Error handling
  - (should) Has to be handled on a hardware-close level
  - Mainly it can be executed here, only in case of fatal errors might be handled on higher levels.

I/O software goals II.

- Synchronous (blocking), asynchronous (interrupting) transfering mode are supported.
  - In the synchronous mode it is easier to write user programs.
  - Operating system have to support asynchron mode as well.
- Buffering
  - Everybody use it e.g. keyboard buffer etc.
- Device usage in distributed or monopole mode
  - The disk can be used by several processes in the same time.
  - A magnetic tape or CD-writer is used in a monopol way.

The structure of I/O software system

- The structure consists of layers
  - Typically it is organized into 4 layers.
- Hardware device
  1. Interrupt handling layer
     - Handled on the lowest level of kernel.
     - Protected the critical section (or the whole) by semaphore block.
  2. Devicedriver programs
  3. Device independent operating system program
  4. User I/O device-using program

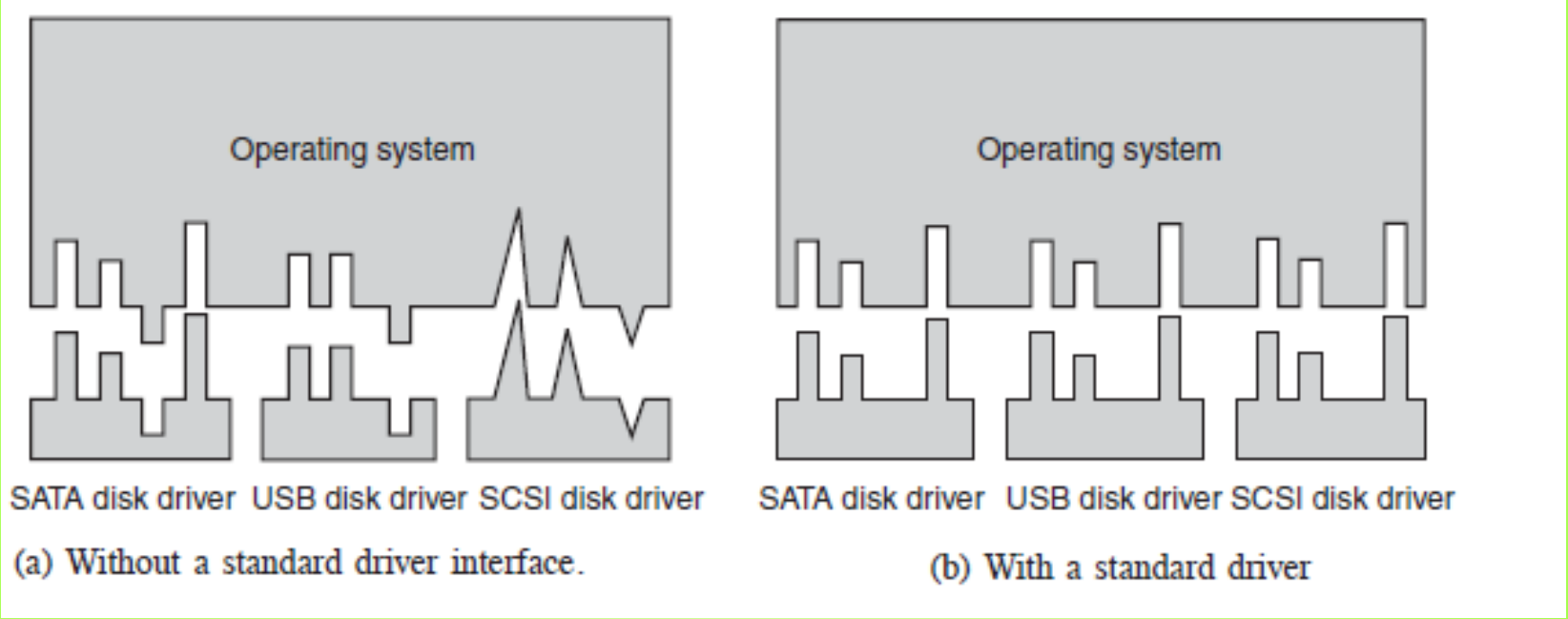
Device driver programs

- Device specific code – driver
- This knows punctually the features of the device
  - Mouse driver tells the shift, which button was pressed.
  - Disk driver knows the moving of the heads, the settings for a given sector, track.
  - Etc.
- Its task is to serve the abstract request arriving from the above level
  - Simple device, one request in the same time
  - Intelligent ones, can be accept several demands simultaneously (scsi)
- Handles the device through I/O ports, using interrupt handling.
- Block-type – character-type devices

The task of a device independent I/O program I.

- To support a standard interface.
  - The same naming, calling conventions.
  - To connect the symbolic names of I/O devices to real drivers.
    - Majaor device number: identifier of the driver
    - Minor device number: + parameter, e.g.. Indicating write-read
  - Protection of I/O device
    - Applying filesystem-like permissions.

Standard interface – illustration.



The task of a device independent I/O program II.

- Buffering similar to adapters, general idea the user is independent from the device independent I/O program. (It writes quicker into the buffer than to the „device”.)
- Error handling
- To reserve and release monopol mode devices
- To form device independent block size.
  - Logical block size of the disks

User I/O programs

- Two categories of library I/O subroutines
  - Transmit parameters to the system call
    - N=write(fd, buffer, db);
  - Work some task then system call
    - printf(„%d apple”, n);
- Spooling
  - Handling mode of monopol devices (printer)
  - Special processes handle the storage device, spooling directories. Demons.

Usage of monopol mode resources

- As we saw formerly they are an important group of I/O devices.
- Handling of system internal table is of the kind (e.g. process table).
- Only one process can use it in the same time.
  - E.g: To print 2 files „parallel” is not the really.
- Race condition occures not only in the case of monopol I/O devices.
  - They are at simple memory sections too, but typically at monopol I/O devices.
- Typical situation: two processes wait the same thing.
  - Two polite men before the elevator – the elevator go they stay…

Deadlock

- Two or more processes – during to allocate the same resource – blocks each other in further execution.
  - Punctual definition: A process set is in a deadlock if each of the process waits for an event which can be caused by another process.
- It is not connected only to I/O devices
  - Parallel systems
  - Databases
  - Etc.

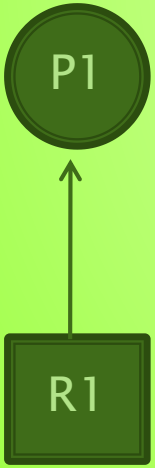
Deadlock conditions

- Coffman E.G.: 4 conditions are needed for evolution of deadlocks
  1. Mutual exclusion: Each resource is assigned to 1 process or free.
  2. Holding and waiting. A process possessing a resource can ask for another one.
  3. No preemption. A resource can be released only by the process holding it.
  4. Circular wait. Two or more process-chain evolve in which all of the processes wait for a resource holding by another one.

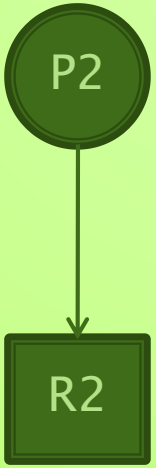
Graph model of deadlock

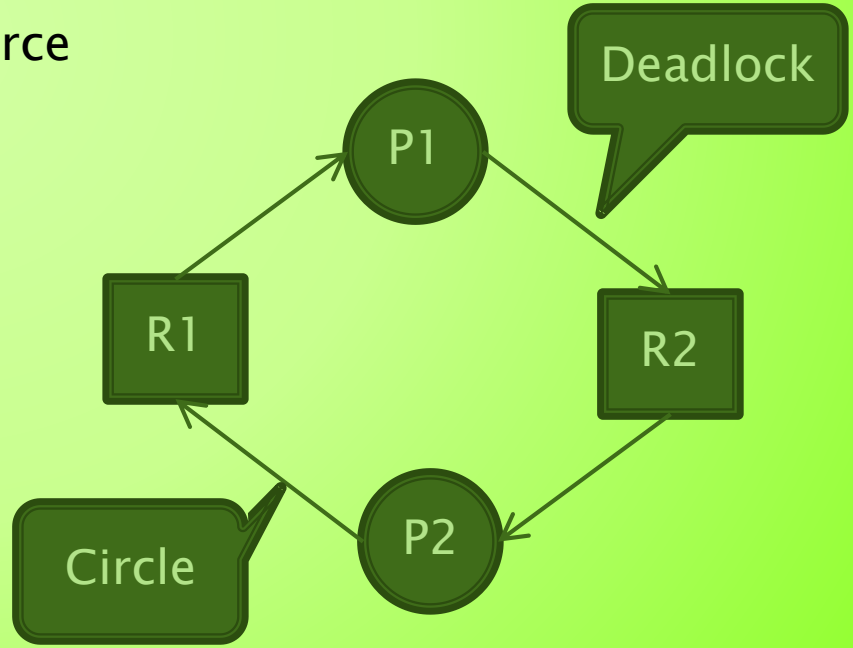
- Modeling deadlock conditions using graphs (Holt, 1972)
  - Process- circle
  - Resource- square
  - Finding a circle in the graph of resources, processes it means a deadlock.

Holding a resource

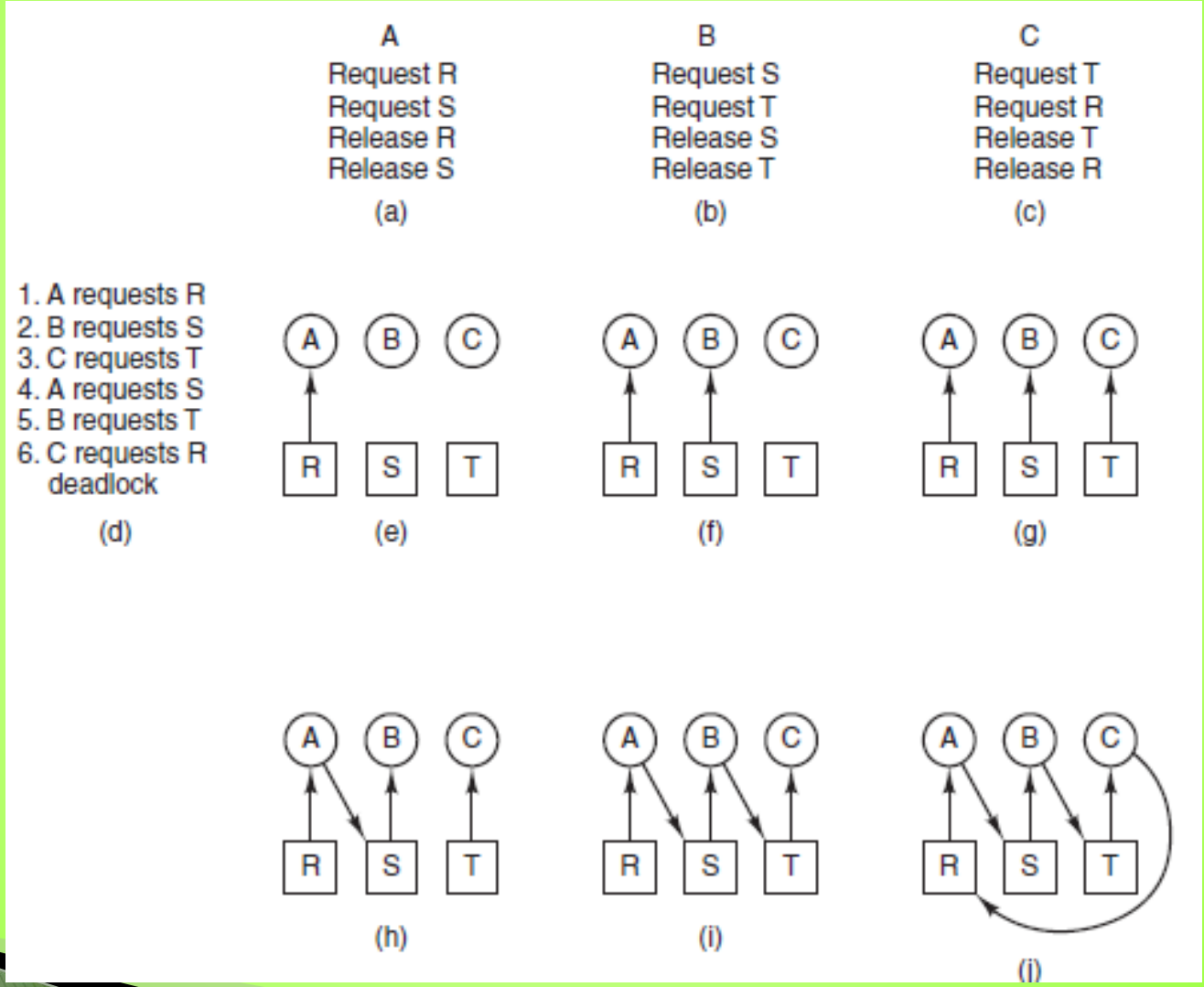


Asking for a resource





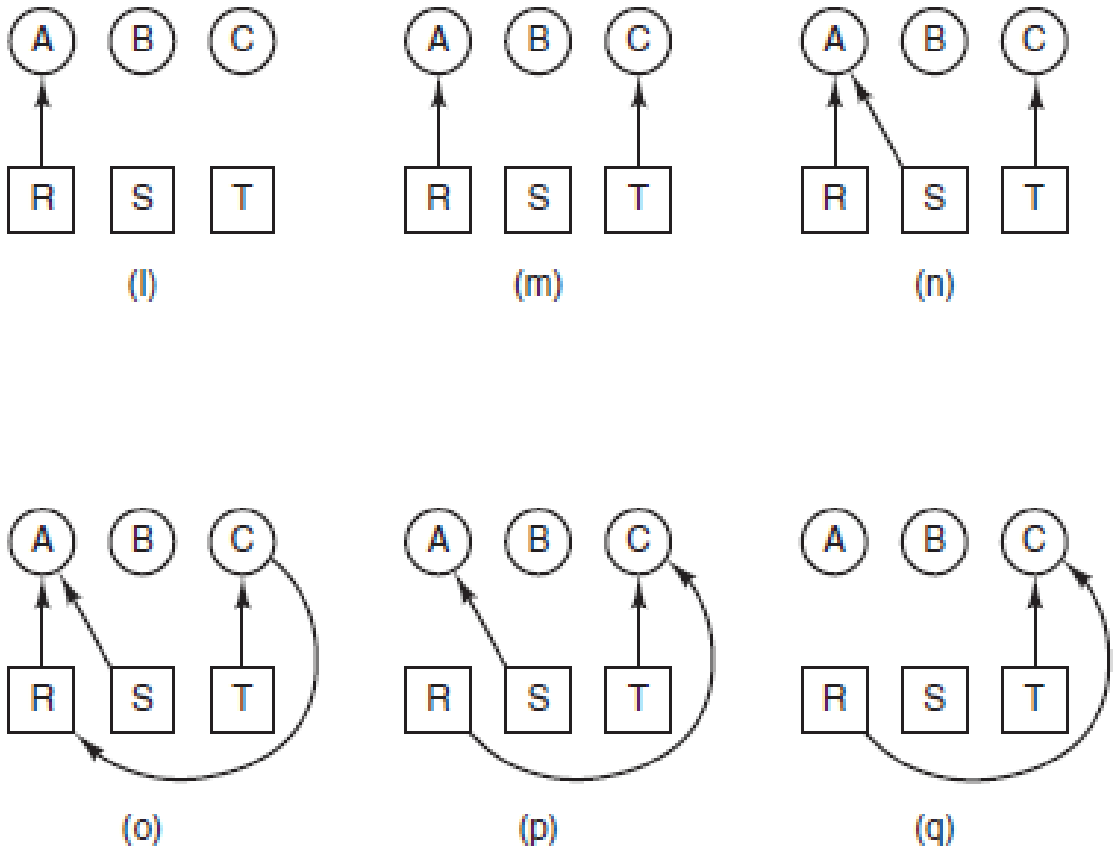
Evolving a deadlock, example



Avoiding deadlock, example

1. A requests R
2. C requests T
3. A requests S
4. C requests R
5. A releases R
6. A releases S
   no deadlock

(k)



Deadlock strategy

1. Ignoring the problem.
   - We do not care with it hoping that it will not find us, if still…
2. Detection (recognizing and restoring).
   - We let to appear the deadlock (circle), we recognize it and take action.
3. Preventing. To disappoint one of the 4 needed conditions.
4. Dynamical avoidance. Resources allocation „carefully”.

1. Ignoring the problem

- This method is known also as „ostrich algorithm”.
- The question is, what does it mean, and how frequent this problem is?
- According to the studies the rate of deadlock problems and other crashes (compiler, operating systems, hw, sw error) is 1:250.
- The Unix, Windows also use this „method”.
  - The price is too big for the probable profit.

2. Recognizing and restoring

- We are monitoring the resource demands and releases continously.
- We handle the resource-allocation graphs permanently.
  - If somewhere there is a circle we cease one of the processes from the circle.
- Other method: we do not pay attention to the resource graph at all, instead of it a process is blocked more than x (half an hour?) time, we terminate it.
  - This method is known in the case of mainframes.

3. Preventing

- Always there is a constraint for one of the 4 Coffman conditions.
  - Mutual exclusion. If a single resource is never assigned to alone 1 process, there is no deadlock!
    - It is uneasy, while at the usage of a printer – the printer demon solves the problem, but the printer buffer is a disk area where a deadlock may evolve.
  - If there is no such situation in which the process is holding the resources and is waiting for further resources there is no deadlock either. We can achieve this in two ways.
    - We have to know the resource request of the process forward.
    - If a process wants a resource it has to release all of its possessed ones.

3. Preventing (cont.)

- The third Coffman condition is the no preemption. To avoid this is rather difficult.
  - During a printing process it is not favored to give the printer to another process.
- The fourth condition, the cyclical waiting can be ceased easier.
  - A simple method: Each process can possess only 1 resource in the same time.
  - Other method: Assign ordenal numbers to the resources and processes may request the resources due to this ordering.
    - It is a good preventing method, but there is no proper order!

4. Dynamical avoidance

- Is there a method with which we can avoid the deadlocks?
  - Yes there is, if some informations (resource) are known forward.
- banker's algorithm (Dijkstra,1965)
  - Similar to a small town banker's practice in trusting.
- Safe states, such situations in which there is a starting state range which result each process can allocate the demanded resources and finish!
- The banker's algorithm examines at the appearing of each request it checks whether the fulfillment of a request results a safe state or not?
  - If it is so, then the request is validated otherwise it is delayed.
  - Originally it was planned only for 1 resource.

Banker' algorithm, example states (1 resource)

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

(a)

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

(b)

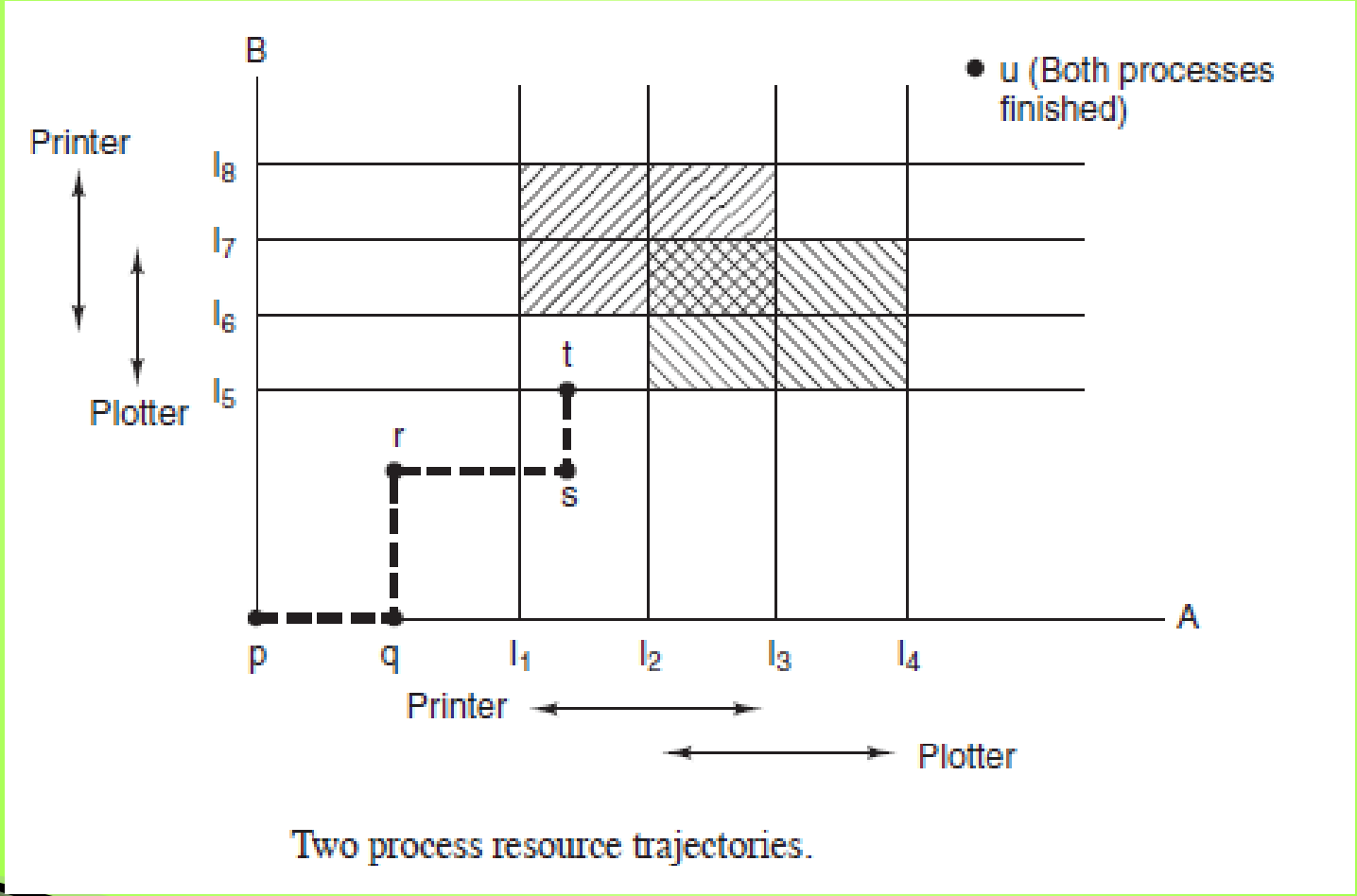
	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

(c)

Three resource allocation states: (a) Safe. (b) Safe. (c) Unsafe.

Two process resource trajectory



Two process resource trajectories.

Banker's algorithm, in the case of several type resources

- We use the 1 resource idea:
  - Notation: F(i,j) the allocation of the i. of the process j. resource
  - M(i,j) is the request of the i. process for the j. resource
  - E(j) is all of the resources being at service.
  - S(j) is the free resources being at service.

1. Search for i queue where M(i,j)<=S(j), if there is not such possibility then there is a deadlock, because no process can finish its running.
2. The i. process gets everything, it finishes then we add its allocated resources to S(j).
3. Repeat 1,2 steps while they end or there is a deadlock.

Banker's example for several resources

Process	Tape drives	Plotters	Printers	Blu-rays
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Printers	Blu-rays
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still assigned

E = (6342)
P = (5322)
A = (1020)

B can allocate 1 printer this will be a safe state. (D can end then A,E,C,B.)

Banker's algorithm summary

- The previous preventing and this also asks such informations (the punctional resource requests, the number of processes), which are difficult to give forward.
  - Processes come into existance dynamically, resources modify dynamically.
- That is why it is used rarely in practice.
- …

Thanks for your attention!

zoltan.illes@elte.hu