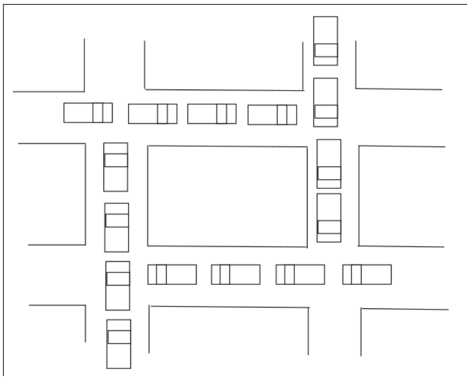
Contents

1 Worksheet 5

1. Consider the following deadlock:



   (a) Show that the four necessary conditions needed for deadlock hold in the traffic dead lock.

   The four necessary conditions for a deadlock are (1) mutual exclu- sion; (2) hold-and-wait; (3) no preemption; and (4) circular wait. The mutual exclusion condition holds as only one car can occupy a space in the roadway. Hold-and-wait occurs where a car holds onto their place in the roadway while they wait to advance in 47 the roadway. A car cannot be removed (i.e. preempted) from its position in the roadway. Lastly, there is indeed a circular wait as each car is waiting for a subsequent car to advance. The circular wait condition is also easily observed from the graphic.

   (b) Provide a simple rule for avoiding deadlock in the example.

A simple rule that would avoid this traffic deadlock is that a car may not advance into an intersection if it is clear they will not be able to immediately clear the intersection.

2. List three overall strategies in handling deadlocks.

3. Can we break 'mutual exclusion' condition to prevent deadlock? Can we break 'hold and wait' condition? Justify your answer.

4. In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

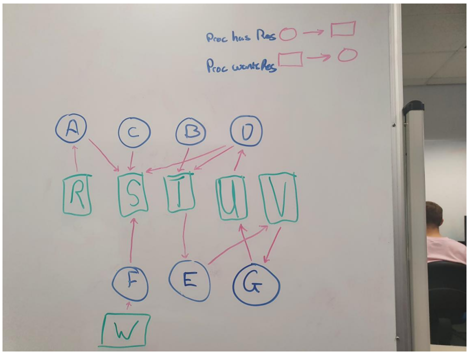
   - Increase Available (new resources added).
     This could safely be changed without any problems.
   - Decrease Available (resource permanently removed from system).
     This could have an effect on the system and introduce the possibility of deadlock as the safety of the system assumed there were a certain number of available resources.
   - Increase Max for one process (the process needs more resources than allowed, it may want more).
     This could have an effect on the system and introduce the possibility of deadlock.
   - Decrease Max for one process (the process decides it does not need that many resources).
     This could safely be changed without any problems.
   - Increase the number of processes.
     This could be allowed assuming that resources were allocated to the new process(es) such that the system does not enter an unsafe state.
   - Decrease the number of processes.
     This could safely be changed without any problems.

5. Consider a system with seven processes, A through G, and six resources, R through W, each with one instance. Resource ownership is as follows.

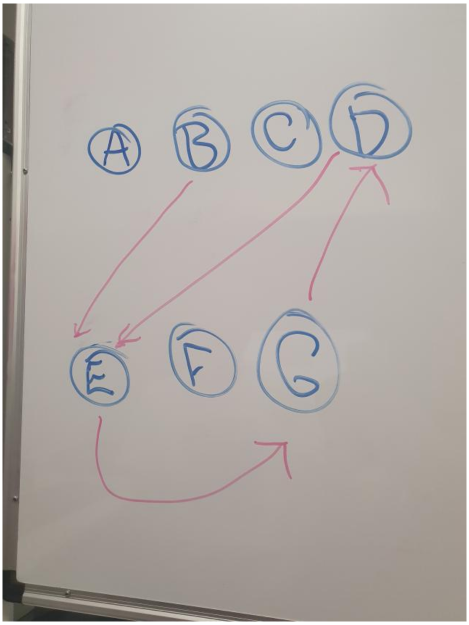
   - Process A holds R and wants S
   - Process B holds nothing but wants T
   - Process C holds nothing but wants S
   - Process D holds U and wants S and T
   - Process E holds T and wants V

- Process F holds W and wants S
- Process G holds V and wants U

(a) Draw resource-allocation graph for the system



(b) Draw the corresponding wait-for graph



(c) Is this system deadlocked?

Yes, the system is deadlocked.

(d) If so, which processes are involved?

The processes involved are D,E,G

6. Consider the following snapshot of a system that is using the banker's algorithm:

	Allocation				Max			
	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2
P_1	1	0	0	0	1	7	5	0
P_2	1	3	5	4	2	3	5	6
P_3	0	6	3	2	0	6	5	2
P_4	0	0	1	4	0	6	5	5

Available			
A	B	C	D
1	5	2	0

(a) What is the content of matrix Need?

	Need			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	1

(b) Is the system in a safe state?

Yes! $P_0 \rightarrow P_3 \rightarrow P_2 \rightarrow P_1 \rightarrow P_4$

(c) If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately?

Yes! $P_0 \rightarrow P_3 \rightarrow P_4 \rightarrow P_2 \rightarrow P_1$

7. Consider a system with the following resource types:

- tape drives (4 units)
- plotters (2 units)
- printers (3 units)
- CD ROMs (1 unit)

At time t, there are three processes with the following information for their resource allocations and additional resource requests:

- process 1: allocation - one printer, additional requests – two tape drives and one CD ROM

- process 2: allocation - two tape drives and a CD ROM, additional requests – one tape drive and one printer
- process 3: allocation - a plotter and two printers, additional request - two tape drives, one CD ROM, and one plotter

(a) Is the system deadlocked? (Show the process sequence if the system is not deadlocked, and show the processes involved if the system is deadlocked)

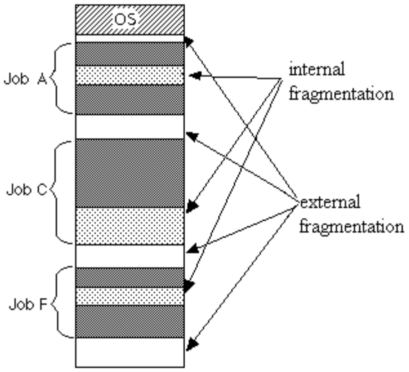
The system is deadlocked due to P_1 and P_3

(b) Is the system deadlocked if process 3‘s additional requests include only two tape drives and one plotter?

This successfully resolves the deadlock $P_0 \rightarrow P_3 \rightarrow P_2 \rightarrow P_1$

2 Worksheet 6

1. Explain the different between internal and external fragmentation.



2. What are the advantages of using paging? What is contained in the page table?

   paging is a method of minimising fragmentation. It involves including

3. Most systems allow programs to allocate more memory to its address space during execution. Data allocated in the heap segments of programs are an example of such allocated memory. What is required to support dynamic memory allocation in the following schemes:

   - contiguous-memory allocation
   - pure segmentation
   - pure paging

4. Consider a computer with 16-bit logical address, and a page size of 4K. How many bits are there for the page number and for the offset number? How many pages are there?

5. Given memory partitions of 100K, 500K, 200K, 300K, and 600K (in order),

   (a) How would each of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212K, 417K, 112K, and 426K (in order)?

   (b) Which algorithm makes the most efficient use of memory?

   (c) Show how compaction can be done on each

6. 6. Consider a paging system with the page table stored in memory.

   (a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

   (b) If we add associative registers, and 75% of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there)

   (c) Consider a logical address space of eight pages of 1024 words each, mapped onto a physical

7. Consider a logical address space of eight pages of 1024 words each, mapped onto a physical memory of 32 frames

   (a) How many bits are there in the logical address?
   To answer this we need to calculate the amount of space available in the logical address: $8 \equiv 2^3$ *pages of* 2^{10}

   $$2^3 \cdot 2^{10} = 2^{13}$$

   13 *bits*

   (b) How many bits are there in the physical address?
   The total number of bits in the physical address is given by:

   $$2^5 \cdot 2^{10} = 2^{15}$$

   15 *bits*

8. Considering the following segment table, what are the physical addresses of the following logical addresses?

   (a) 0, 430

   (b) 1, 10

   (c) 2, 500

   (d) 3, 400

   (e) 4, 112

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

3 Worksheet 7

1. Briefly describe the following terms:

   - Virtual Memory
     Utilises secondary storage to act as an extension to main memory. Pages can be stored in virtual memory when the system is experiencing a high load. Secondary storage allows for processes which are not being used to not take up space in main memory.
   - Thrashing
     When a process does not have all of its frames in memory it may cause page faults thus loading in more pages in from secondary storage. This results in low CPU utilisation while the pages are loaded. If the CPU then increases the level of multiprogramming to compensate the same will happen again. This is known as thrashing.
   - Demand Paging
     Bringing an entire process (not just a page) into memory only when it is needed & advantageous for the computer. (less IO, less memory, faster response, more users)
   - Page Fault
     Occurs when a page is requested but it is not found in main memory. It is necessary to load the page from virtual memory which takes additional time.
   - Page Replacement
     The process which is followed to move a frame which is currently not in use to virtual memory, swapping it with another page which is stored in virtual memory.
   - Modify/dirty bit
     Modify or dirty bit indicates the page under observation has been modified and needs to be stored back on the virtual memory disk to allow the process which owns it to retrieve it later. If the modified bit is not set then the process has not used the page and it does not need to be copied into swap space as there is already a valid copy there.
   - Vaild/invalid bit
     Indicates if the current page has been loaded from the disk or not. If it has been loaded from the disk then it is valid. If not then it is invalid and needs to be

loaded again.

- Belady's anomaly.
  Belady's anomaly is the anomaly observed when increasing the number of frames for a given process. The assumption is that more frames means more hits and therefore improved performance. When in fact the opposite is true.
- Prepaging
  refers to a operating system loading the page working set back into main memory to prevent an influx of page faults when context switching (this would cause low CPU utilisation)
- Working set model
  The working set model is a model which describes the most recently used pages for a program. there are a number of ways to figure this out but the generally accepted way is to define a window Δ which is the number of past requests to review in order to determine which pages should be included in the working set.
- Lock bit
  A lock bit is present on a frame. it indicates that that frame cannot be swapped into virtual memory. an example of this may be the driving code for an operating system as performance would be severely affected if it had to load this from virtual memory every few minutes.

2. When do page faults occur? Describe the actions taken by the OS when a page fault occurs
   a page fault occurs when a page which was previously loaded in to memory was subsequentially swapped to the backing store and the was paged again. The process the Operating system follows for dealing with a page fault is as follows:

   (a) OS checks if the reference is valid, if invalid terminate with an error
   (b) otherwise find a free frame using the free frame list
   (c) schedule a disk read operation to read the page into the target frames
   (d) modify the page table appropriately
   (e) restart instruction which was interrupted by page fault.

3. A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456H. Explain how the system establishes the corresponding physical location.
   The virtual address in binary form is: 0001 0001 0001 0010 0011 0100 0101 0110 Since the page size is $2^{12} \equiv 4kB$, the page table size is 2^{20}. Therefore the low order 12 bits "0100 0101 0110" are used as the displacement into the page, while the remaining 20 bits "0001 0001 0001 0010 0011" are used as the displacement in the page table to locate the frame.

4. Suppose we have a demand-paged memory. The page table is held in registers. It takes 8ms to service a page faults if an empty page is available or the replaced page is not modified, and 20ms if the replaced page is modified. Memory access time is 100ns. Assume that the page to be replaced is modified 70% of the time. What is the maximum acceptable page fault rate for an effective access time of no more than 200ns?

$$EAT(\alpha) = \alpha \cdot \Big((8ms) \cdot 30\% + (20ms) \cdot 70\% \Big) + (1-\alpha) \cdot 100ns$$

$$.2ms = \alpha \cdot \Big((8ms) \cdot 30\% + (20ms) \cdot 70\% \Big) + .0001ms - \alpha \cdot .0001ms$$

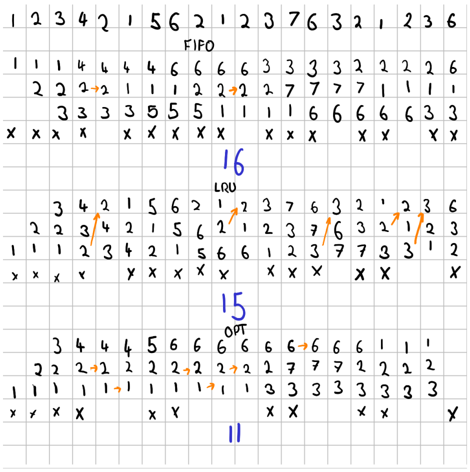
$$.0001ms = \alpha \cdot \Big((8ms) \cdot 30\% + (20ms) \cdot 70\% - .0001ms \Big)$$

$$\alpha = \frac{.0001ms}{\Big((8ms) \cdot 30\% + (20ms) \cdot 70\% - .0001ms \Big)} = \frac{100}{163999} = 6.0975 \cdot 10^{-4}\%$$

5. Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember that all frames are initially empty, so your first unique pages will all cost one fault each.



1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
									FIFO										
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3
X	X	X	X		X	X	X	X	X		X	X	X		X	X		X	X
									16										
									LRU										
		3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
	2	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
1	1	1	2	3	4	2	1	5	6	6	1	2	3	7	7	3	3	1	2
X	X	X	X		X	X	X	X	X		X	X	X		X	X			X
									15										
									OPT										
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	
X	X	X	X			X	X				X	X			X	X			X
									11										

(a) LRU replacement

(b) FIFO replacement

(c) Optimal Replacement

6. What cause thrashing? How does the system detect thrashing? Once it detects thrashing what can the system do to eliminate thrashing?

7. Consider a demand-paging system with a paging disk that has an average access and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory with an access time of 1 microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference, if the page-table entry is in associative memory. Assume that 80% of the accesses are in the associative memory, and that, of the remaining 10% (or 2% of the total) cause page faults. What is the effective access time?

8. An OS supports a paged virtual memory, using a central processor with a cycle time of 1 microsecond. It costs an additional 1 microsecond to access a page other than the current one. Pages have 1000 words, and the paging device is a drum that rotates at 3000 RPM and transfers 1 million words per second. The following statistical measurements were obtained from the system:

   - 1 percent of all instructions executed accessed a page other than the current page.
   - Of the instructions that accessed another page, 80 percent accessed a page already in memory.
   - When a new page was required, the replaced page was modified 50 percent of the time.

   Calculate the effective instruction time on the system, assuming that the system is running one process only, and that the processor is idle during drum transfers.

   (a) Time per page to controller:

$$10^6 \cdot \frac{word}{sec} \cdot \frac{1}{10^3} \frac{page}{word} = 10^3 \cdot \frac{page}{sec}$$

$$T_a = \frac{1}{10^3} \frac{sec}{page} = 1000 \frac{\mu s}{page}$$

   (b) Time per page to/from disk:

$$\frac{3000}{1} \cdot \frac{rev}{min} \cdot \frac{1}{60} \cdot \frac{min}{sec} = 50 \frac{rev}{sec} = \frac{1}{50} \cdot \frac{sec}{rev}$$

$$T_b = \mu = \frac{0.02}{2} = 0.01 \frac{sec}{page}$$

   (c) effective read write time per page to disk

$$T_t = T_a + T_b = 11000 \frac{\mu s}{page}$$

(d) *same page (one memory access)* 99%

$$T_1 = 99\% \cdot 1\mu s = .99\mu s$$

(e) ***different page*** (1%)

i. *In memory (two memory accesses)* (80%)

$$T_2 = 1\% \cdot 80\% \cdot (2\mu s) = 0.016\mu s$$

ii. *Not in memory* (20%)

A. *Victim needs replacement (2 disk operations)* (50%)

$$T_3 = 1\% \cdot 20\% \cdot 50\% \cdot (11000 \cdot 2\mu s) = 22\mu s$$

B. *Victim does not need replacement (1 disk operation)* (50%)

$$T_4 = 1\% \cdot 20\% \cdot 50\% \cdot (11000\mu s) = 11\mu s$$

(f) *Result*

$$T = 34.006\mu s$$

9. Consider a demand-paged computer system where the degree of multiprogramming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization?

   (a) CPU utilization 13%; disk utilization 97%

   (b) CPU utilization 87%; disk utilization 3%

   (c) CPU utilization 13%; disk utilization 3%

4 Worksheet 8

1. The open-file table is used to maintain information about files that are currently open. Should the operating system maintain a separate table for each user or just maintain one table that contains references to files that are being accessed by all users at the current time? If the same file is being accessed by two different programs or users, should there be separate entries in the open file table?

2. Sequential access can simulate direct access, and direct access can also simulate sequential access. Which simulation is more efficient? Justify your answer.

3. In two-level directory, how do users access system files?

4. Consider a system where free space is kept in a free-space list. Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer. Suggest a scheme to ensure that the pointer is never lost as a result of memory failure.

5. Consider a system that supports the strategies of contiguous, linked, and indexed allocation. What criteria should be used in deciding which strategy is best utilized for a particular file?

6. Consider a file currently consisting of 200 blocks. Assume that the file control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single level) allocation strategies, if for one block, the following conditions hold. In the contiguous- allocation case, assume that there is no room to grow in the beginning, but there is room to grow in the end. Assume that the block information to be added is stored in memory.

   (a) The block is added at the beginning
   (b) The block is added at the middle
   (c) The block is added at the end
   (d) The block is removed from the beginning
   (e) The block is removed from the middle
   (f) The block is removed from the end

7. Why must the bit-map for file allocation be kept on mass storage, rather than in main memory?

8. Consider a file system on a disk that has both logical and physical block size of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

   (a) How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long)
   (b) If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

5 Workshop 9

1. State the advantages and disadvantages of placing functionality in a device controller rather than in the kernel.

2. How does DMA increase system concurrency?

3. Describe circumstances under which blocking I/O should be used

4. Why not just implement non-blocking I/O and have processes busy-wait until their device is ready?

5. When are caches useful? What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching, why not make it that large and eliminate the device?

6. What is the main different between a cache and a buffer?

7. For each of the following I/O scenarios, would you design the OS to use buffering, spooling, caching, or a combination? Would you use polled I/O, or interrupt driven I/O? Give reasons for your choices

   (a) A mouse is used with a graphical user interface

   (b) A tape drive on a multitasking OS (assume no device pre-allocations is available)

   (c) A disk drive containing user files

   (d) a graphics card with direct bus connection, accessible through memory mapped I/O.

8. Why might a system use interrupt driven I/O to manage a single serial port, while polling I/O to manage a front-end processor, such as a terminal concentrator?

9. Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order is

   86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

   Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

   (a) FCFS

   (b) SSTF

   (c) SCAN

   (d) LOOK

   (e) C-SCAN

   (f) C-LOOK

10. Explain why SSTF scheduling tends to favour middle cylinders over the innermost and outermost cylinders.

11. Briefly explain the differences among the six levels for RAID, showing how each improve reliability, access time, and I/O rate.

12. RAID level 3 is able to correct single-bit errors using only one parity-drive. What is the point of RAID level 2?

6 Worksheet 10

1. What is the need to know principle? Why is it important for a protection system to adhere to this principle?

2. Why is the separation of mechanism and policy a desirable property?

3. What are the main differences between capability lists and access lists?

4. Capability lists are usually kept within the address space of the user. How does the system ensure that the user cannot modify the contents of the list?

5. Buffer-overflow attacks can be avoided by adopting a better programming methodology or by using special hardware support. Discuss these solutions.

6. Attacks from inside the system (by somebody that has already been logged in, legally or illegally) can be, among the few, in the forms of Trojan Horses, Login Spoofing, Logic Bombs, and trap doors. Explain how each work, and ways to prevent them, if possible.

7. What is the purpose of using a "salt" along with the user-provided password? Where should the "salt" be stored, and how should it be used?

8. When a file is removed, its blocks are generally put back on the free list, but they are not erased. Do you think it would be a good idea to have the OS erases each block before releasing it? Consider both security and performance factors in your answer, and explain the effect of each.