COMP3301: 2022 Exam solutions

UQAttic

Get more out of your study time. Join UQAttic's revision chat. Or Slack, if you're a better man.

Other exam papers

Please **contribute** to these documents.

> If you're looking for an effective way to familiarize yourself with the course material, you can't go past collaborating with fellow students. We have labored to put these up, and so at the very least point out where you think we are wrong!
>
> You'll get more out of the course, you'll do better in the exam, and other students will benefit from your input as well.
>
> To get editing permissions, simply go to the chatroom and provide us with your Google Account address.e

Style.

> Type answers in blue beneath each question.
>
> If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.
>
> If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

Communicate.

> Head over to uqattic.net and click "Chat Now!". You'll find a chatroom full of students just like you. Talk about a revision document (like this one) or swap prep tips. If you have your own IRC client, point it to irc.uqattic.net, port 6667, channel #attic.

Question 1. (20 marks)

Calculate performance metrics for various process schedulers.

All of the schedulers are pre-emptive – a process is pre-empted as soon as a higher priority process appears.

In the following tables, times are all in milliseconds.
Assume processes do not stall for I/O. Ignore scheduler delays.
Processes that miss their deadline are not included in the calculation of individual and average completion time, turnaround time, waiting time or throughput. The CPU time of aborted processes is included in CPU utilisation.

(a) Complete the following tables for an Earliest Deadline First (EDF) Scheduler.

Process Number	Arrival Time	Execution Time	Deadline	Completion Time	Turnaround Time	Waiting Time
P1	0	4	7	4	4	0
P2	5	4	15	10	5	1
P3	6	1	11	7	1	0
P4	10	4	14	14	4	0
P5	13	1	30	19	6	5
P6	14	4	25	18	4	0

Average Waiting Time	1ms
Average Turnaround Time	4ms
Actual CPU Utilisation	94.74% (sum(cpu time used))/(num cores * total time) or total cpu runtime - idle/total cpu runtime = 18/19 x 100 = 94.74%
Average Throughput	315 jobs/s

Process Timeline Table. Mark where a process Starts (S), Runs (R), Pauses (P), Ends (E). You can add more columns if required.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
P1	SR	R	R	R	E															
P2						SR	P	R	R	R	E									
P3							SR	E												
P4											SR	R	R	R	E					
P5																			SR	E

P6															SR	R	R	R	E	

(b) Repeat part a) but now assume that two CPU cores (core 1 and core 2) can be used to run processes, concurrently. Complete the following tables for an Earliest Deadline First (EDF) Scheduler. You must consider the core affinity (core 1, core 2 or X – any core). You can assume that a process with no affinity (X) can run on any core.

Process Number	Arrival Time	Execution Time	Deadline	Core Affinity	Completion Time	Turnaround Time	Waiting Time
P1	0	4	7	1	4	4	0
P2	5	4	15	2	9	4	0
P3	6	1	11	X	7	1	0
P4	10	4	14	2	14	4	0
P5	13	1	30	1	14	1	0
P6	14	4	25	X	18	4	0

Average Waiting Time	0ms
Average Turnaround Time	3ms
Actual CPU Utilisation	50% + 2 I think this is wrong :o no it's right You should count all the SR and R from both Easiest way to calculate it is to sum the execution times and divided by the greatest completion time + 1 (* cpu cores), so (4 + 4 + 1 + 4 + 1 + 4) = 18, Greatest completion time is 18, + 1 for the end of the process, then the equation is 18 / (18 * 2) = 50%
Average Throughput	333 jobs/s completed job/ greatest cpu runtime 6/18 x 1000 = 333

Process Timeline Table (Mark where a process Starts (S), Runs (R), Pauses (P), Ends (E). You can add more columns if required.

Core 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
P1	SR	R	R	R	E															
P5														SR	E					
P3							SR	E												
P6																				

Core 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
P2						SR	R	R	R	E										
P4											SR	R	R	R	E					
P6															SR	R	R	R	E	

Question 2. (20 marks)

Consider a file system where disk block addresses are represented by a 32- bit integer. A disk block is **2048** bytes long.

Assume that a single command (read or write) to the disk controller consists of a starting disk block address, the number of contiguous blocks to read/write, and the location of the memory buffer for the data. Assume that commands can be sent continuously to the controller (you don't need to wait until the previous one has been serviced unless the next read needs the contents of the previous block). Assume that disk requests are serviced in the order received and with a latency of **5ms** between each request (from the end of the previous data transfer or from the time the command was received, whichever is longer), and a streaming transfer rate of one block per **50µs**.

Now, consider a direct access file currently consisting of the following blocks, listed below, and calculate the requested information for each particular file system below. Assume that CPU operation times can be neglected. Assume that the whole file can fit into memory if needed.

Disk Blocks used for a file:
50, 78, 96, 101, 234, 321, 456, 678, 789, 890, 1234, 2340, 3459, 5623, 8990, 12056, 23560, 37450, 78900

(a) Index allocation File System
Assumptions – the file control block is in memory, the file index allocation table is in memory, any new data that needs to be written to the file is in memory. Assume the disk blocks in the file are not contiguous.

Calculate the number of disk controller commands, the number of disk block I/O operations (i.e. total number of disk block reads and writes), and the total time to complete each of the following operations.

Disk Blocks used for a file (copied from previous page for convenience): 50, 78, 96, 101, 234, 321, 456, 678, 789, 890, 1234, 2340, 3459, 5623, 8990, 12056, 23560, 37450, 78900

	Number of disk controller commands	Number of disk block I/O operations	Total Time (ms)
(i) One block is added to the end of the file.	1	1	5.05
(ii) One block is added to the beginning of the file.	1	1	5.05
(iii) Data in location 8066 (bytes, from top of file) is altered.	1 2	1 2	5.05 10.1
(iv) One block is removed from the end of the file.	0 1	0	0 5
(v) Block 890 is removed from the file.	0 1	0	0 5
(vi) The file is deleted (data does not need to be overwritten).	0 1	0	0 5

For iii) since we aren't writing the whole block (we're starting at offset 1922 of block 3), won't we have to read in the block then write it back with the altered data? Because it says a command has a "starting block address" but not an offset into the block, so i don't think we can be granular enough to start writing mid block. This is also assuming we are writing < 126 bytes, otherwise we cross over to block 4 [+2]

^posted qn on ed

But why is removal 0??
"*Assumptions – the file control block is in memory, the file index allocation table is in memory"* all we need to do is remove the entry from the index table in-memory and we can assume that it will be flushed to disk at a later time.

Total time = time between request (= 5ms) * request time (= M) time + time per block (= 50 micro secs / 1000 = 0.05ms) * block count (= N)
= 5*M + 0.05*N

Can someone give me a resource to answer this sort of question? Im confused.

TRIM uses 1 controller command but no I/O, for time see #965 (2024 cohort). Tutor said it all looks good.

(b) Linked File System

Assumptions – the file control block is in memory, each file block contains a link to the next file block, any new data that needs to be written to the file is in memory. Assume the disk blocks in the file are not contiguous.

Calculate the number of disk controller commands, the number of disk block I/O operations (i.e. total number of disk block reads and writes), and the total time to complete each of the following operations.

Disk Blocks used for a file (copied from previous page for convenience): 50, 78, 96, 101, 234, 321, 456, 678, 789, 890, 1234, 2340, 3459, 5623, 8990, 12056, 23560, 37450, 78900

	Number of disk controller commands	Number of disk block I/O operations	Total Time (ms)c
(i) One block is added to the end of the file.	3	3	15.15
(ii) One block is added to the beginning of the file.	1	1	5.05
(iii) Data in location 8066 (bytes, from top of file) is altered.	4 5 [+3]	4 5 [+3]	20.2 25.25 [+3]
(iv) One block is removed from the end of the file.	19 20	19	95.95 105.95 100.95? +1
(v) Block 890 is removed from the file.	11 12	11	55.55 60.55
(vi) The file is deleted (data does not need to be overwritten).	1	0	0 5

Why is i) 3 commands? Is it because read last block, write last block with new end block, write new end block? Doesnt make sense to me because shouldnt adding and removing end block should be the same?

Apparently for (iv) we need to read up to the second to last block(37450) then write it as the new end block which makes it 18R+1W while in (i) we just read the last block, write the new block, update the new end block 1R+2W

To do delete operations (for iv and v) you would still need to send a message to the device controller to say that the block is ready for erasure (TRIM command).for trim do we still need to count the 5ms time as well? Actually, probably not because you can just send a forget #965 seems to say otherwise tho - the guy put in 5ms but the tutor said it all looks good yeah fair enough

iii) Assuming our blocks are 0-indexed (i.e. block 50 is block 0 of our file), so therefore block 101 is block 3 of our file.

How does this change how many blocks you need to find block 101? 50->78->96->101 + write to 101

Read B0 (0-2047), read B1(2048-4095), read B2 (4096-6143), read B3(6144-8191), write B3. it doesn't

matter what the block nums are but it is 5 I/Os But why do you need to read B3? Do you need to read a block before you write to it? I just posted on ed about Q2a(iii) asking that
bc we are only partially editing the block, we gotta read to know what the start of it (bytes 6144-8065) was i think. otherwise that's not just altering from byte 8066, we wouldn't know that the rest was to write it back the same. This is all assuming we can only write whole blocks at a time.

v)why we need to read 890? i think we only read before 890 and modify the link of 789, that takes 9R and 1W
Read block 890 to get the link to block 1234<-silly me, i got it X)

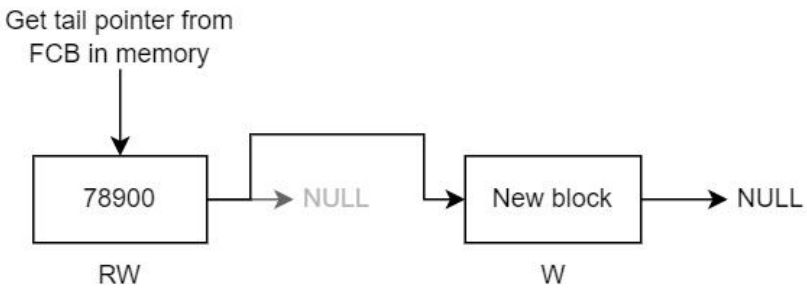
Wtf is this question :-| BEYOND COOKED (LOL i think this could be on this years exam)

In linked, if you're removing a block and it's the 10th block, then you do 11 I/Os, it's actually quite simple. **All removals have one extra disk controller command.**

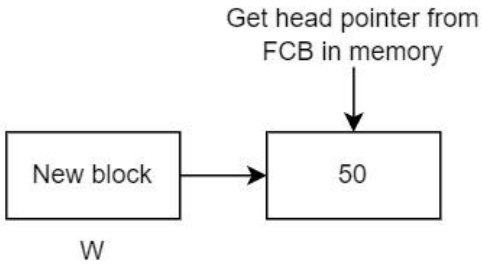
They have to supply sample answers with workings for the future students ffs

Goodluck everyone

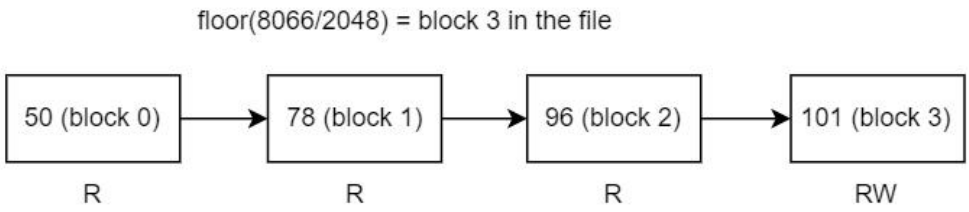
i)



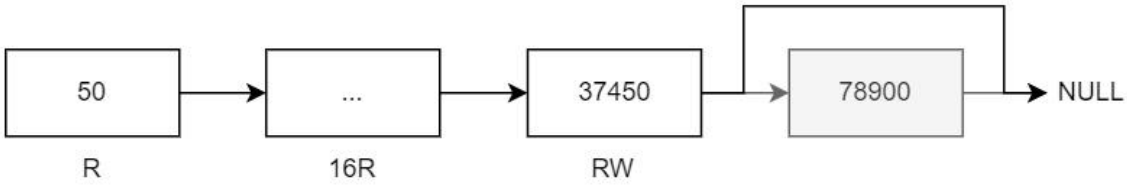
ii)



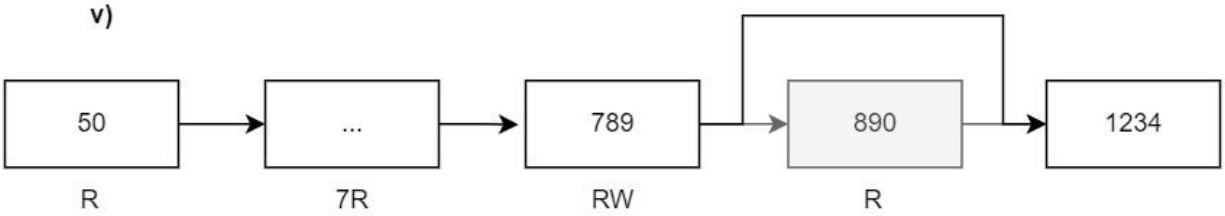
iii)



iv)



v)



Question 2 (continued)

(c) Free Space Map
You can assume the last disk block in the file is the last disk block of the disk volume.
Disk Blocks used for a file (copied from previous page for convenience): 50, 78, 96, 101, '234, 321, 456, 678, 789, 890, 1234, 2340, 3459, 5623, 8990, 12056, 23560, 37450, 78900

(i) Calculate the minimum memory (in Kbytes – assume 1K is 1024 bytes) required to store the bitmap of free space blocks of the disk volume?

$\frac{78900\ block}{8\ blocks/byte} \times 2^{-10} KB = 9.63KB$

Assuming 0 indexing there are 78901 blocks

Do we need to round up to a block boundary? Making the answer 10KB?

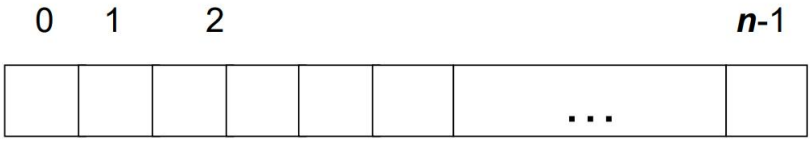
Prolly better not to.

/2

(ii) Show a subset of the first 8 bits of the free space bitmap of the file.

1	1	1	1	1	1	1	1

1 = free
0 = used



bit[*i*] = { 1 ⇒ block[*i*] free; 0 ⇒ block[*i*] occupied

/2

(iii) Calculate the minimum memory (in disk sectors – assume sector is 2048 bytes and a pointer is 32 bits long) required to store a linked list of the free space blocks?

$$\#\ free\ blocks\ =\ 78900\ -\ 19\ =\ 78881$$

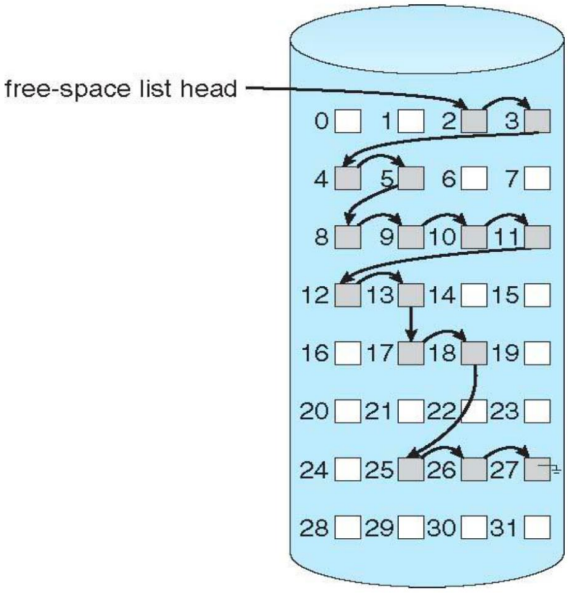
$$\#\ pointers\ per\ sector\ =\ \frac{2048}{4} - 1\ =\ 511\ pointers$$

$$\#\ sectors\ =\ ceil\left(\frac{78881}{511}\right) = 155\ sectors$$

/2

Ok but i think this question is kinda misleading, bc a linked list isn't gonna just be a bunch of pointers, that wouldn't be linked, that's just a list. All that u will need to store is the list head right? Everything else is stored inside the free blocks. I may be wrong.

- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
  - No need to traverse the entire list (if # free blocks recorded)



(iv) Which file allocation method is best for keeping track of free space and explain why?

With linked list allocation we needed 310 KB compared to bitmap which only needed 9.63 KB. Therefore bitmap is better since it uses less space to keep track of the free space blocks. +1

File allocation methods are how you assign blocks to files, no? So you want indexed/linked list/contiguous as an answer. I said contiguous as it is most likely to maximize the size and minimise the count of groups of free memory blocks, allowing you to use grouping/counting methods of free space tracking -1

In this context, we want to track free space. The red one focuses on file allocation methods.

/2

Question 3. (20 marks)

The following memory pages are accessed:

1,2,3,1,2,3,0,1,2,4,4,5,5,1,2,1,2,0,1,7,3,1

(a) Calculate the number of page faults if the First In First Out (FIFO) page replacement algorithm is used, with **i)** 3 page buffer and **ii)** 5 page buffer. Also state what the minimum buffer size is to ensure that no page faults occur.

i) 14 page faults
ii) 10 page faults
With demand paging there will always be page faults when we access the page for the first time and put it onto the buffer. – If we assume that our buffer contains all pages already, then it is 7.

(b) Calculate the number of page faults if the Least Recently Used (LRU) page replacement algorithm is used, with **i)** 3 page buffer and **ii)** 5 page buffer. Also state what the minimum buffer size is to ensure that no page faults occur.

i) 13 page faults
ii) 8 page faults
With demand paging there will always be page faults when we access the page for the first time and put it onto the buffer. – If we assume that our buffer contains all pages already, then it is 7.

(c) List four different methods of implementing the Least Recently Used (LRU) page replacement algorithm.

1) Two stacks
2) Counter
3) Hash table
4) Linked list

1. Counter method
2. Stack method
3. Reference bit (approximation)
4. Advancea (approximation)

(d) Explain when a page is considered a 'victim' and what must occur?

A page is a victim when the page replacement algorithm selects it to be evicted from the buffer and replaced with a new page.
As for what must occur, before evicting check the dirty bit, if it's dirty (has been edited) write it back into memory.

(e) What is Copy-on-Write? And what variation of the fork function uses it?

COW is when multiple things reference the same memory for reading, but when one of them tries to write to the memory it is first copied and diverges from the shared memory.
vfork() uses COW for the initial parent-child set up. [-1]
fork() uses COW for the initial parent-child set up. vfork does not I just went with what the lectures said.
I think the lecture slides just havent been updated in a while, since from man vfork
"vfork() was originally used to create new processes without fully copying
the address space of the old process, which is horrendously inefficient
in a paged environment. It was useful when the purpose of fork(2) would
have been to create a new system context for an execve(2). Since fork(2)

is now efficient, even in the above case, the need for vfork() has diminished."
It also says "That sharing of memory was removed in 4.4BSD", so it doesnt even COW anymore lol.

Are you sure?

`vfork()` variation on `fork()` system call has parent suspend and child using copy-on-write address space of parent

- Designed to have child call `exec()`
- Very efficient

So should we answer vfork on the exam if asked?
Textbook says fork is CoW, vfork is not. This is true for Linux and most BSDs (OpenBSD is actually special in that both are CoW). Lecture slides are directly contradicting the textbook…. No, read the man page for Vfork
Vfork is the CoW version - “Thus, for greater efficiency, BSD introduced the vfork() system call, which did not fully copy the address space of the parent process, but borrowed the parent's memory and thread of control until a call to execve(2) or an exit occurred.” - read the man page for Vfork
Yeah but fork is CoW now though lol
I guess if the exam shows this question i’ll just eat my pen then +1

Answer is fork #974 on ed
“...so the answer to this question should be `fork()`...” +3

Absolute cinema
Ill just pee in the exam hall
I'll poop in the urinals

Question 4. (20 marks)

(a) What are four benefits of multi-threaded architectures?

- Responsiveness - a multithreaded application that delivers an interface (visual) to the user, but has time-consuming operations that must occur, but shouldn't slow down the interface.
- Resource sharing - By default, threads share memory and the resources of the process they belong to, which allows several different threads access the same address space.
- Scalability - a single threaded application can only run on one processor at a time, however with multiple threads they can be allocated to different processors for concurrent execution.
- Economy - since threads share the resources of the process they belong to, it is more economical (time and memory wise) to create and context-switch threads rather than to create additional processes.

(b) For the following multi-threading models, briefly describe each model.

Many to One)

Many to one details that each kernel-level thread is mapped to many user-level threads.

One to One)

One-to-one details that each user-level thread is mapped to a kernel-level thread.

Have to be careful not to create too many user threads, as will hurt performance (textbook p167).

Many to Many)

Many-to-many details that each user-level thread can be mapped against many kernel-level threads (not exclusive to a single thread)

With many-to-many you don't have to worry about creating too many threads, as something else schedules the user threads on the kernel threads for the programmer (textbook p167).

(c) For the previous multi-threading models, state which model is best for concurrency and why?

One-to-one is the best for concurrency, largely due to allowing another thread to run when a thread makes a blocking system call, and also allows multiple threads to run in parallel using multiple processors.

Concurrency is managing multiple processes using limited resources, so one to many is the best for concurrency not one to oneOne to many - Limited concurrency because all user threads must sha

re one kernel thread

Which one is the right one then?
One-to-One is best for concurrency because:

- Provides true parallel execution
- Direct kernel scheduling
- No blocking dependencies between threads
- Can fully utilize multiple processors
- Each thread can run independently

From reading the textbook it appears many-to-many is the best, having all the

advantages of one-to-one, without the drawback of the programmer having to worry about using too many threads (using too many threads with one-to-one can cause performance issues).

The textbook also says that many-to-many is much harder to implement, and the problems with 1-to-1 are becoming smaller as with modern CPU's with their higher number of processors, limiting kernel threads is becoming less important (the main problem with 1 to 1), and this is why Windows/Linux still use 1-to-1, so it's actually not so straightforward. Classic better in theory worse in practice

/2

(d) What is the difference between a user thread and a kernel thread?

/2

User-level threads are generated created by users, whereas kernel threads are supported and managed directly by the operating system.

(e) What is a thread library? Also list two ways of implementing it.

/3

A thread library is an API provided to a programmer for creating and managing threads.
Two ways of implementing a thread library is as follows:

- Provide a library entirely in user space with no kernel support
- Implement a kernel-level library supported directly by the operating system.

(f) What is a hardware thread and state an advantage? Also state the type of CPU architecture that supports hardware threads.

/3

A hardware thread details a thread of execution on physical hardware (CPU) that can be switched to if another hardware thread (on the same physical hardware - CPU) stalls for any reason. Only one hardware thread can run at a time on a single CPU. The inclusion of multiple hardware threads on a single is sometimes called hyper-threading which is supported by contemporary intel processors
A hardware thread (also called a logical processor) is a CPU's hardware-level ability to maintain multiple execution states and switch between them efficiently - essentially appearing as multiple "logical" processors to the operating system.

Advantage:

- When one thread is stalled (e.g., waiting for memory), the CPU can quickly switch to another thread without the overhead of a full context switch, improving CPU utilization and overall throughput

Architecture that supports this:

- Simultaneous Multithreading (SMT)
- Intel's implementation is called Hyper-Threading
- Each physical core appears as 2 or more logical cores to the OS

Total ____/ 20

Question 5 (20 marks)

(a) In an OS, what four groups can Input/Output (IO) devices be grouped into? Also give an example application or peripheral (e.g. USB stick) of each type.

i) Block device - Hard drive

ii) Character device - Keyboard

iii) Memory-mapped IO - Graphics card

iv) Network device (sockets) - Bluetooth

/8

(b) Which two IO device registers can be accessed using the c function ioctl()?
Control register and data-out register.

/2

(c) For each IO operation listed below, give a brief explanation and also state a disadvantage.

i) Blocking

Blocking is when your process has to wait for an IO request to finish before continuing. For example it might have to wait for a blocking IO request such as reading from a file.

A disadvantage to blocking is that your process cannot do anything else while it is waiting for the request.

ii) Nonblocking

Nonblocking is when the IO request returns as much as available without blocking.

A disadvantage of nonblocking is it could require busy-waiting to poll for the completion of the full IO request.

iii) Asynchronous

The IO request is serviced in the background and an interrupt is generated once it is completed.

A disadvantage is that is complex and difficult to use.

/6

(d) For each Kernel IO subsystem listed below, give a brief explanation and also give an example application (e.g. CCID).

i) Caching

Caching is when a faster device holds a copy of the data for faster access.
Example: TLB.

ii) Spooling

Spooling is when we hold the output for a device until it is ready to receive it. This is necessary if the device can only serve one request at a time.
Example: Printer.

/4

Question 6. (20 marks)

You have been asked to design an operating system for the new Alset self-driving electric vehicle. The vehicle uses a smart card (with CCID) as a key to lock/unlock the doors and start the vehicle. The vehicle is equipped with realtime parking cameras (high priority and 1ms latency), LIDAR (Laser) scanner for realtime driving (high priority and 1s latency), GPS used for navigation (low priority and 60s latency) and a video display for entertainment (low priority and 1ms latency). The vehicle also comes with a built-in solar panel for power charging and must be kept in direct sunlight. This leads to extreme temperature conditions that causes errors in various electronic devices such as hard drive storage and affect the running of CPU cores.

You can also assume the following:

- The operating systems uses multiple CPU cores for redundancy.
- Magnetic disk mass storage is used, to save cost.
- A list of authorised users is kept in a file on the onboard file system. • A large video library (stored as files) must be kept on board for the user's selection.
- The vehicle can run for 4 hours before requiring charging.

For each of the indicated aspects of the operating system:

(i) Describe your suggested design alternative,

(ii) Give one other less favourable choice,

(iii) Explain two advantages of the suggested alternative compared to the less favourable choice

(a) What type of Parallelism should be used (Data vs Task)? (i) Suggested solution (ii) Less favoured alternative (iii) Two advantages of suggested solution
(i) Task (ii) Data (iii) • Task Parallelism involves distributing not data but tasks (threads) across multiple computing cores, where each thread is performing a unique operation (e.g. process GPS, scan with laser for real-time driving, display video, etc). Data Parallelism would only be preferred when handling large datasets that must be processed. • Task Parallelism is safe as critical functions such as realtime driving can be run on a single core being unaffected (largely) by entertainment options running times (e.g. displaying a video). /5

../Question 6 continued over the page

Question 6 (continued)

(b) What type of CPU scheduler should be used?
(i) Suggested solution (ii) Less favoured alternative
(iii) Two advantages of suggested solution

(i) Multi-level queues
(ii) Round-robin
(iii)

- Multi-level queues allows the higher priority tasks to go first (parking cameras, laser scanner) regularly which is critical in this application to ensure safety features.
- User can determine correct ordering based on likelihood to cause (different levels of) damage:
  - Laser scanner
  - Parking camera
  - Smart card
  - Gps
  - Entertainment

less favored could also be 'Earliest Deadline First'

/5

../Question 6 continued over the page

Question 6 (continued)

(c) What security mechanism should be used to ensure that only authorised users can access the vehicle?
(i) Suggested solution (ii) Less favoured alternative
(iii) Two advantages of suggested solution

i) biometrics
ii) NFC Card
iii) 1. Easy to add new people 2. Hard (but not impossible) to lose a finger/eye

/5

"The vehicle uses a smart card (with CCID) as a key to lock/unlock the doors and start the vehicle." seems like there is already a security mechanism in place?

i) Multi-Factor Authentication
ii) Single-Factor Authentication
iii) -Even if one factor is compromised (key fob copied/PIN observed), vehicle remains secure as other factors are needed
-Can remotely disable specific factors if compromised

./Question 6 continued over the page

(d) What RAID Disk level (0 to 6) should be used?
(i) Suggested solution (ii) Less favoured alternative
(iii) Two advantages of suggested solution

(i) RAID level 6
(ii) RAID level 0

(iii)

- Stores extra redundant information to guard against multiple drive failures, which may be desired due to extreme temperatures.
- As compared to RAID level 0, which has no error-parity bits or redundant information, this level has decreased chances of crucial data loss
- Increases mean time to data loss
- Increases mean time to repair
- Increases mean time to failure

/5

Total ____/ 20

BLANK PAGE FOR WORKING END OF EXAMINATION