# COMP 346: Operating Systems
# Final Exam

Duc Nguyen

*Gina Cody School of Computer Science and Software Engineering  Concordia University, Montreal, QC, Canada*

Winter 2020

## Contents

**1 Question 1** **4**
1.1 Problem: . . . . . . 4
1.2 Part A . . . . . . 4
1.2.1 Best CPU utilization . . . . . . 4
1.2.2 Throughput if 2 processes are running . . . . . . 6
1.3 Part B . . . . . . 6

**2 Question 2** **7**
2.1 Problem: . . . . . . 7
2.2 launch(P(0), P(1)) . . . . . . 7
2.2.1 Mutual exclusion . . . . . . 7
2.2.2 Good progress . . . . . . 8
2.2.3 Deadlock-free . . . . . . 8
2.3 launch(P(0), P(1), P(2)) . . . . . . 9
2.3.1 Mutual exclusion . . . . . . 9
2.3.2 Good progress . . . . . . 10
2.3.3 Deadlock-free . . . . . . 11

**3 Question 3** **15**
3.1 Problem: . . . . . . 15
3.2 Support team . . . . . . 15
3.3 Reasons to support . . . . . . 15
3.4 Reasons to reject . . . . . . 15

**4 Question 4** **17**
4.1 Multiple threads provides an advantage over single thread . . 17
4.1.1 Best case scenario . . . . . . 17
4.1.2 Average case scenario . . . . . . 17
4.1.3 Worst case scenario . . . . . . 18
4.1.4 Conclusion . . . . . . 18
4.2 Example showing benefit of multi-threading in a single CPU environment . . . . . . 18

**5 Question 5** **19**

**6 Question 6** **20**
6.1 Linked lists allocation for allocated blocks . . . . . . . . . . . 20
6.2 Linked lists allocation for unallocated blocks . . . . . . . . . . 20

**7 Question 7** **21**
7.1 Problem: . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
7.2 Choose side for argument about SSTF disk scheduling . . . . . 21
7.3 Reasons behind 2 supporting and rejecting SSTF . . . . . . . 21
7.3.1 Supporting SSTF disk scheduling . . . . . . . . . . . . 21
7.3.2 Rejecting SSTF disk scheduling . . . . . . . . . . . . 22
7.4 New constraints apply . . . . . . . . . . . . . . . . . . . . . 22

**8 Question 8** **23**
8.1 Implementation of Belady Optimal Algorithm . . . . . . . . . 23
8.2 Implementation of LRU algorithm . . . . . . . . . . . . . . . 23
8.3 Internal fragmentation . . . . . . . . . . . . . . . . . . . . . 24

**9 Question 9** **25**
9.1 Problem: . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
9.2 Average effective access time for a page . . . . . . . . . . . . 25
9.3 Average effective access time for a page if page is NOT in main memory . . . . . . . . . . . . . . . . . . . . . . . . . . 26

**10 Technology and Market Impact on Operating Systems** **27**
10.1 Problem: . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
10.2 Technology trends in the past 5 years . . . . . . . . . . . . . . 28
10.2.1 IoT (Internet of Things) . . . . . . . . . . . . . . . . . 28
10.2.2 CPU processing (speed, capacity, cores, etc.) . . . . . . 29
10.2.3 Main Memory (i.e. RAM) . . . . . . . . . . . . . . . . 29
10.2.4 Cloud Computing . . . . . . . . . . . . . . . . . . . . . 30
10.2.5 Big-data Processing . . . . . . . . . . . . . . . . . . . . 31
10.3 Technology trends anticipate in the next 5 years . . . . . . . . 32
10.3.1 5G Networks . . . . . . . . . . . . . . . . . . . . . . . . 32
10.3.2 Artificial intelligence . . . . . . . . . . . . . . . . . . . 32
10.3.3 Main Memory (i.e. RAM) . . . . . . . . . . . . . . . . 33
10.3.4 Reacting to world pandemics, such as the current COVID-19 . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
10.3.5 Fault-tolerance . . . . . . . . . . . . . . . . . . . . . . . 34

# 1 Question 1

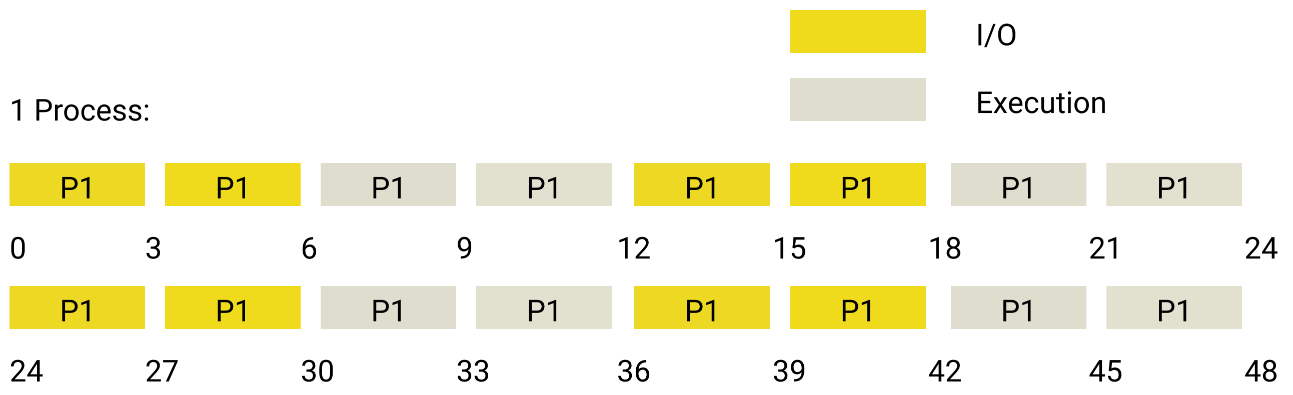
## 1.1 Problem:

Assume that all processes in the system have similar behavior, where a process spends ¡ of its time performing I/O and ¡ of the time performing execution. The system is capable of overlapping execution time for a process with I/O time of another. The system is utilizing also RR for scheduling with a time quantum of 3ms (ignore context switch time; i.e. assume it takes 0 ms). Show the Gantt Chart for all your answers.

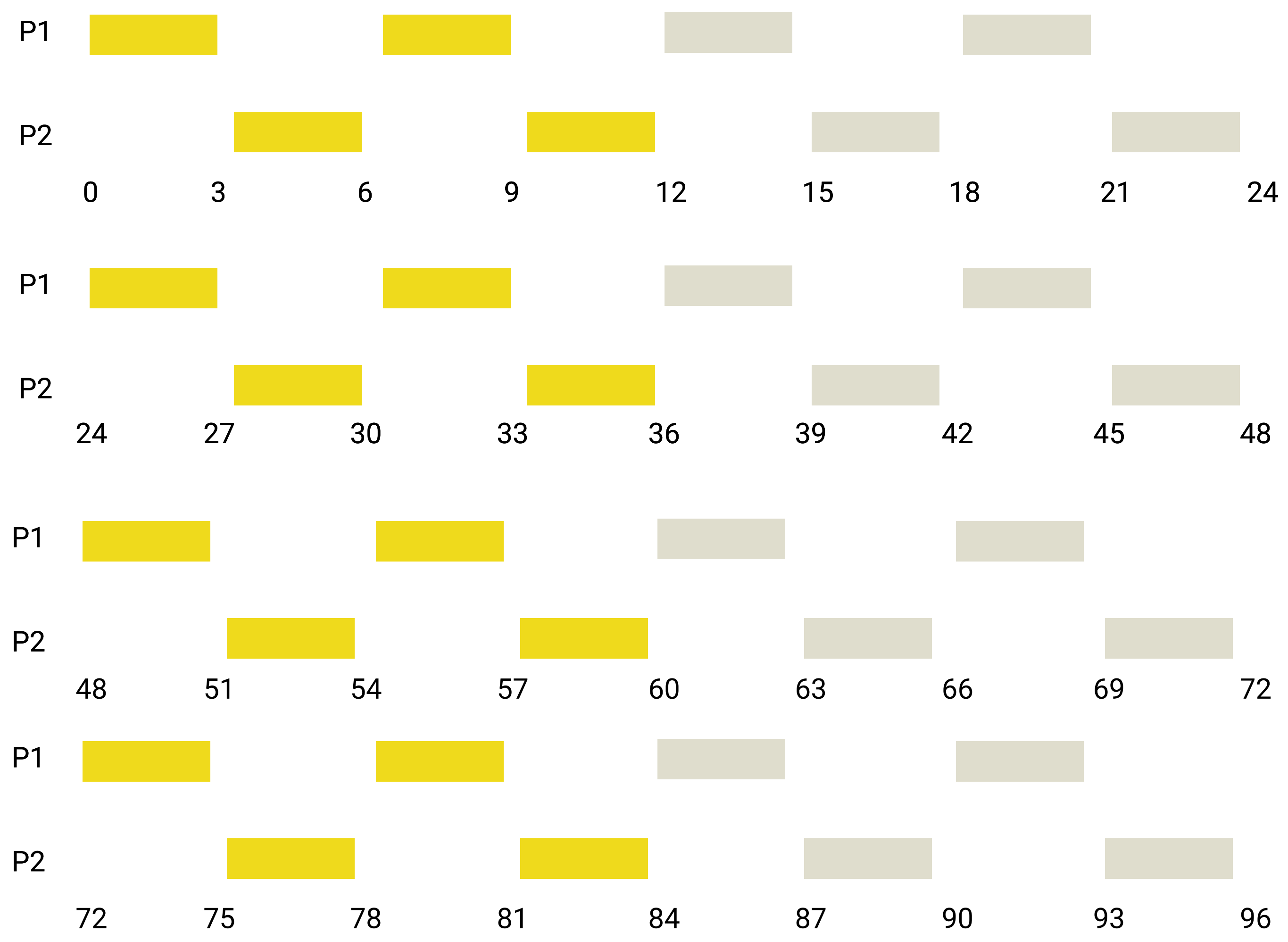
## 1.2 Part A

### 1.2.1 Best CPU utilization

1 Process:



2 Processes:



- With 1 process running, CPU utilization = $\frac{24}{48} * 100\% = 50\%$
- With 2 processes running, CPU utilization = $\frac{48}{96} * 100\% = 50\%$

#### 1.2.2 Throughput if 2 processes are running

$$\text{throughput} = \frac{2}{96} \approx 0.02083 \text{ (process)}$$

## 1.3 Part B

## 2 Question 2

### 2.1 Problem:

Given the following code, where id indicates a process number (or ID), launch() indicates concurrent start of the processes passed on its parameters, and N is an integer representing the number of processes that will be launched (i.e. processes that will run on that system).

```
boolean blocked[N];
for(int i = 0; i < N; i++)
 blocked[i] = false.
int turn = 0;
int other;
P(int id)
{
while(true)
{
 blocked[id] = true;
 while(turn != id)
 {
if(id == 0 || id == 2)
 other = 1;
else
 other = 0;
 while(blocked[other]) { turn = other;}
turn = id;
 }
 /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
 blocked[id] = false;
}
}
```

### 2.2 launch(P(0), P(1))

#### 2.2.1 Mutual exclusion

The program achieves mutual exclusion since there is no possible outcome in which more than 1 thread enters the Critical Session at once.

### 2.2.2 Good progress

The program achieves good progress since it manages to continue execution on time without any delay.

### 2.2.3 Deadlock-free

The program is deadlock-free since there is no possible outcome in which deadlock happens and fails the solution.

## 2.3 launch(P(0), P(1), P(2))

### 2.3.1 Mutual exclusion

The program will not achieve **mutual exclusion**. One scenario is as following:

```
 //.. Running P(0)
P(int id)
{
    while(true)
    {
         blocked[id] = true;
         while(turn != id)
         {
            // .. By pass the while loop because turn = id
         }
         /* > > > > > > Interruption happens < < < < < < */
         /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
         blocked[id] = false;
    }
}
```

The current global variables are as following:

```
/* global values at the time */
blocked[0] = true
blocked[1] = false
blocked[2] = false
turn = 0
other = null
```

The process P(2) starts:

```
// Continue from the interruption of P(0)
//.. Running P(2)
P(int id) {
    while(true) {
         blocked[id] = true; // blocked[2] = true
         while(turn != id)
         {
            if(id == 0 || id == 2)
            other = 1;     // other = 1
           else
            other = 0;
            while(blocked[other]) { turn = other;} // By pass the
                while loop as blocked[1] = false
           turn = id;     // turn = 2
     }
     /* EXECUTE CRITICAL SESSION AT THE SAME TIME AS P(0) */
     /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
     blocked[id] = false;
    }
}
```

**Explaination:**
As the scenario can happen, P(1) and P(2) might get into the critical session at the same time; thus, the code is not mutual exclusion.

### 2.3.2 Good progress

If there is no deadlock, the program would achieve good progress since it continues to run without delays

### 2.3.3 Deadlock-free

Consider the scenario:

```
 //.. Running P(0)
P(int id)
{
    while(true)
    {
         blocked[id] = true;
         while(turn != id)
         {
            // .. By pass the while loop because turn = id
         }
         /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
         /* > > > > > > Interrupted by P(1) < < < < < < */
         blocked[id] = false;
    }
}
```

The current global variables are as following:

```
/* global values at the time */
blocked[0] = true
blocked[1] = false
blocked[2] = false
turn = 0
other = null
```

P(1) starts:

```
// Continue from the interruption of P(0)
//.. Running P(1)
P(int id) {
    while(true) {
         blocked[id] = true; // blocked[1] = true
         while(turn != id)
         {
            if(id == 0 || id == 2)
             other = 1;
            else
             other = 0;     // other = 0
            /* > > > > > > Interrupted by P(2) < < < < < < */
             while(blocked[other]) { turn = other;}
            turn = id;
      }
      /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
      blocked[id] = false;
    }
}
```

The current global variables are as following:

```
/* global values at the time */
blocked[0] = true
blocked[1] = true
blocked[2] = false
turn = 0
other = 0
```

P(2) starts:

```
// Continue from the interruption of P(1)
//.. Running P(2)
P(int id) {
    while(true) {
         blocked[id] = true; // blocked[2] = true
         while(turn != id)
         {
           if(id == 0 || id == 2)
            other = 1;     // other = 1
           else
            other = 0;
           /* > > > > > > Interrupted by P(1) < < < < < < */
            while(blocked[other]) { turn = other;}
           turn = id;
     }
     /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
     blocked[id] = false;
    }
}
```

The current global variables are as following:

```
/* global values at the time */
blocked[0] = true
blocked[1] = true
blocked[2] = true
turn = 0
other = 1
```

P(1) resumes:

```
// Resume from the interruption in P(2)
//.. Running P(1)
P(int id) {
    while(true) {
         blocked[id] = true;
         while(turn != id)
         {
         // .. .. ..
            /* > > > > > > Resume from the last in terruption < < <
                < < < */
             while(blocked[other]) { turn = other;} // turn = 1 < -
                 - - DEAD-LOCK
            turn = id;
     }
     /* <<<< CRITICAL SECTION IS EXECUTED HERE>>>>>> */
     blocked[id] = false;
    }
}
```

The current global variables are as following:

```
/* global values at the time */
blocked[0] = true
blocked[1] = true
blocked[2] = true
turn = 1
other = 1
```

**Explaination**

As a matter of fact, the while loop in P(1) will start the dead lock.

- If P(2) resumes from its last interruption, it will run the same while-loop as P(1).
- If P(0) resumes from its last interruption, it will execute the critical session and then get to the same while-loop as P(1) has

This scenario produces a possible dead lock situation of the code. Therefore, the code does not achive deadlock-free if **launch** is called as: **launch(P(0), P(1), P(2))**.

# 3 Question 3

## 3.1 Problem:

As a part of a development team in an Operating Systems company, your team was given the sole responsibility of looking at solutions to the Deadlock problem. After a large research in the subject, here is what your team (after being divided into smaller sub-teams) came up with as possible solutions. Team 1, Team 2 and Team 3 proposed i, ii, and iii, respectively.

- Roll-Back and Random
- Release-All and Restart Differently
- Terminate and Revert to Limited Uni-programming

## 3.2 Support team

The team that I would support is team ii) with the solution to release-all and restart Differently.

## 3.3 Reasons to support

Firstly, it would be simple to kill all the processes that involved in a deadlock. After that, tall the process will be restarted in different order. With a great number of resources and processes, the possible combination is huge; thus, the chance of getting a deadlock is reduced. Although the disadvantage of this solution is that all the partial computations in the previous processes will be discarded. However, the computation can be re computed in the new cycle.

## 3.4 Reasons to reject

- **Roll-Back and Random**: The solution is unpredictable in term of timing since each roll back time, the deadlock detection algorithm has to be executed to find deadlock once again to determine whether or not it should roll back one more time. Resulting in a great time complexity if the number of processes is large.

- **Terminate and Revert to Limited Uni-programming:** The solution has the same risk of encounter a deadlock once again as ii) since the possible combination of resource allocation does not change. However, with Limiter Uni-programming, the processes will execute in a much slower speed, and significantly increases the time it needed to restart and complete all the computations, compared to multiprogramming.

# 4 Question 4

## 4.1 Multiple threads provides an advantage over single thread

**Problem:**
Explain whether using multiple threads would really provide an advantage over using just a single thread. You need to explain and justify your answer for the most optimal solution (best case), as well as for worst case and the average case.

**Answer:**
Let i denote the number of iteration / look-up until the searching operation terminates (found the targeted element or the targeted element does not exist inside the array).
Let m denote the number of elements inside the array.
**Note:**

- Assuming multiple threads with at least 2 threads, each thread searches half of the array.
- Assuming 5ns is the time for a single time of looping. Meaning if the array contains n elements, it takes 5n (ns) to look up the whole array.

### 4.1.1 Best case scenario

Best case scenario happens when the searching element is at the beginning of the array; hence, single thread with sequential search will find it in i = 1 $\implies$ time = $5ns$.
With multiple threads, one of the two threads starts and found the result in i = 1, meaning time = 5ns.

### 4.1.2 Average case scenario

With single thread, average case scenario would be when the searching element is at exactly middle of the array.
$\implies i = \frac{m}{2}$. Thus, the look up time is $\frac{5m}{2}(ns)$
With multiple thread (with the assumptions), the average case scenario is also when the searching element is at the middle of the array.

$\implies i = \frac{m}{2}$ since each thread searches one half of the array. However, using multiple thread with Round-Robin scheduling means the operation has to also accommodate context switch time (1 ms each)
$\implies$ time $= \frac{5m}{2} +$ total context switch time(ns)

### 4.1.3 Worst case scenario

Worst case scenario happens when the targeted element does not exist in the array
Thus, single thread operation will sequentially loop through every single element inside the array. Resulting in a total time of 5n (ns).
While with multiple thread operation, the total time is $\implies$ time $= \frac{5m}{2} +$ total context switch time(ns)

**Note:** It is also noticeable that multiple thread programming can use more than 2 threads, 2 threads are the least number.

### 4.1.4 Conclusion

- In the best scenario, multi threading has the same completed time as single thread.
- In the average scenario, multi threading is slower than single thread the total context switch time. However, this can be reduced if there are more threads covers in the right area of the array.
- In the worst scenario, multi threading shows a much greater performance when the data is huge.

Therefore, using multiple threads would really provide an advantage over using just a single thread.

## 4.2 Example showing benefit of multi-threading in a single CPU environment

Web browser operation is an example of benefits of multi-threading. Web browser often has to several operations at the same time. Multi threading can solve the problem so that each tab or instance of the browser might run different job.

## 5 Question 5

# 6 Question 6

**Problem:** Linked lists allocation can be used to keep track of allocated blocks of a file system, as well as to keep track of unallocated blocks. Assume that the probability of pointer corruption in the system is assumed to be very small, and hence negligible.
For each of these two allocation schemes, and considering the given constraints/conditions of the system, indicate if it is possible to safely and efficiently use linked-list allocation for these two types of allocation? your answer should not exceed $\frac{1}{2}$ a page.

## 6.1 Linked lists allocation for allocated blocks

With the constraints that pointer corruption in the system can be negligible, it is safely to use linked-list allocation for allocated blocks since there will be no situation results in lost data.
However, it is arguably to consider linked list allocation implementation in term of efficiently. Since each storage block can be scattered anywhere on the disk, it would take more time for disk seeks and I/O than other methods for tracing the pointers to locate the requested block of a file. Moreover, linked-list allocation does not support direct access capability. To locate a block, it has to be traversed sequentially from the beginning of the file.

## 6.2 Linked lists allocation for unallocated blocks

In this case, it is safe for linked-list allocation to be used. Each time the system wants to allocate a block, the linked list holding unallocated blocks can release one and move the pointer to the next free block.
However, the efficient disadvantage remains the same as mentioned before, even though the speed is mostly faster than usual access block because the list only has to trace the pointer once. Thus, it is slightly inefficent than other methods.

# 7 Question 7

## 7.1 Problem:

You and your classmates have looked carefully at the SSTF disk scheduling then went through an argument, where some indicated that the scheme will have very negative side effects and so it must not be considered, while the others indicated that it will produce superior performance compared to the other available schemes and hence it should be used.

## 7.2 Choose side for argument about SSTF disk scheduling

**Problem:**
Which side are you with? Explain why? your answer should not exceed $\frac{1}{4}$ a page.

**Answer:**
I agree with the side that saying SSTF disk scheduling have very negative side effects because of :

- Starvation of some requests.
- Overhead.

## 7.3 Reasons behind 2 supporting and rejecting SSTF

**Problem:**
Understanding that technique, explain clearly what are the details/reasons behind both support and rejection by the classmates. your answer should not exceed $\frac{1}{2}$ a page.

**Answer:**

### 7.3.1 Supporting SSTF disk scheduling

- **Better seek time:** SSTF has a much better average seek time, in compared with other algorithms, especially FCFS scheduling since SSTF opts for the shortest seek time first while FCFS will go sequentially.

- **Shorter waiting time:** The request which has shortest time is served first; therefore, the waiting time is minimized.

### 7.3.2 Rejecting SSTF disk scheduling

- **Starvation of some requests** SSTF might lead to potential starvation when it comes to requests that take more seek time than others. SSTF serves the shortest seek time request first, and might miss the requests with higher seek time. Thus, it does not serve every request.

- **Overhead:** SSTF has to know the seek time to every request from its current location in advance to determine which request it should serve first. The mechanism leads to overhead for the calculation.

## 7.4 New constraints apply

**Problem:**
Assume that priority is needed, where the requests should be served on a local area on the disk (that is, particular tracks at a local part of the disk are more important than others and hence they need to execute first). Based on these new constraints, would your answer from (i) above change? Explain clearly why or why not. your answer should not exceed $\frac{1}{2}$ a page.

**Answer:**
My answer from (i) would change to supporting SSTF in this case. Since particular track at a local part of the disk is prioritized, SSTF can use its advantage of serving request with shortest seek time. Thus, once the head reaches a part of the local area, the local requests in that part will be served first as they are closest to the head at that time.

## 8 Question 8

### 8.1 Implementation of Belady Optimal Algorithm

**Problem:**
a. Concerning page replacement; is it possible to implement Belady Optimal algorithm, or even an approximation of it? If yes, explain how this can be done; if no, explain why it is not feasible to implement this algorithm.

**Answer**
It is impossible to implement Belady Optimal Algorithm because there is no way to know ahead the number of page faults will be generated and the number of frames are allocated.

### 8.2 Implementation of LRU algorithm

**Problem:**
b. It is indeed possible to implement LRU algorithm for page replacement. Outline a procedure/algorithm for implementing this algorithm efficiently. If you believe your algorithm is efficient, clearly explain why you think so (give strong supportive arguments). If your algorithm is not efficient, explain why it is not (keep in mind that the question is requesting you to implement an efficient algorithm.)

**Answer:**

```
int capacity = 5; // can be any number
int[] arr = { .. } // The array containing page references

int findNumberOfPageFault(int capacity, int[] arr):
    ArrayList a [capacity];
    int count = 0;
    int numOfPageFaults = 0;

    for(int i : arr): {
        if(a.contains(i) == false) {
            if(a.size == capacity) {
                a.remove(0);  // Remove the element at index 0
                a.add(capacity-1, i) // Add a new element, shifting
                    all the other elements 1 index away
            } else {
                //Found page fault
                a.add(count, i)
                numOfPageFaults++;
                count++;
            }

        } else {
            a.remove(i); // i is the object, not the index
            a.add(a.size, i)
        }
    }
    return numOfPageFaults;
```

## 8.3 Internal fragmentation

**Problem:**
Assume that the size of an executable file of a process is evenly divided by the page size (for example, size is 700 and page size is 100, so you have exactly 7 pages). Assume also that we are not interested in external fragmentation and that LFU algorithm, where 3 frames are allocated for the process, is used. Under these conditions, is there anyway that the system suffers from internal fragmentation? Explain your answer very clearly.

**Answer:**
Since the size of the executable file is evenly divided by the page size, there is no way for system to have internal fragmentation.
Internal fragmentation only happens when there is one page with an odd size than others; therefore, that causes internal fragmentation when it is placed on main memory.

## 9 Question 9

### 9.1 Problem:

Consider a demand-paged system that has a fast associative memory (TLB, or cache). To speed up access to pages, it is desired to place the page table in the cache. However, since the page table of a process is too large, the system included only the first 20% of the page table in the cache, 30% in the main memory, and the rest of the table had to be kept in the secondary memory (i.e. page faults may be needed for the page table itself!). Consider uniform request for each of the pages (i.e. each page is requested more or less the same amount of times). To simplify the calculations, assume that cache reference costs 30 ms (milliseconds), memory access costs 200 ms, and page fault time costs 900 ms if the fault is for the page table, and 2 seconds if the fault is for a page. The page fault time for a page includes exactly the following: obtaining the page from the secondary memory, possible swapping, and updating the page table.

### 9.2 Average effective access time for a page

**Problem:**
What is the average effective access time for a page, if that page is in the main memory?
Explain clearly; you must show all your calculations; and provide the exact final answer.

**Answer:**

- If cache hits, then access cache to look up the TLB
- If cache misses, then try accessing main memory.
- If main memory hits, then access main memory and look up the TLB
- If main memory misses $\implies$ it is a page fault for the page table
- $\alpha$ : Cache hit ratio
- $\beta$ : Main memory hit ratio

$$\begin{aligned}
\text{TLB SEARCH TIME} &= (\alpha) * (\text{Cache access time}) + \\
&(1-\alpha)(\beta * \text{Memory access time} + (1-\beta) * \text{page fault}) \\
&= (0.2)(30) + (1-0.2)[0.3 * 200 + 0.7 * 900] \\
&= 558ms(milliseconds) \\
\implies \text{Average EAT} &= \text{TLB SEARCH TIME} + \text{Memory access time} \\
&= 558 + 200 \\
&= 758ms.
\end{aligned}$$

## 9.3 Average effective access time for a page if page is NOT in main memory

What is the average effective access time for a page, if that page is NOT in the main memory?
Explain clearly; you must show all your calculations; and provide the exact final answer.

Since the page is not in main memory, it leads to a page fault for a page.
*Note:* 2 seconds = 2000ms

$$\begin{aligned}
\implies \text{Average EAT} &= \text{TLB SEARCH TIME} + \text{Page fault time} \\
&= 558 + 2000 \\
&= 2558ms.
\end{aligned}$$

# 10 Technology and Market Impact on Operating Systems

## 10.1 Problem:

Some years back, operating systems played a critical rule in changing worldwide technologies and the market, and consequently drove what trended at that time. While this may be true as well today, it is also true that technology trends are also changing and impacting operating systems in many different ways. In this question, you will need to use the concepts learned in class to go beyond what was covered. In other words, you will need to use what you have learned and understood in class to conduct a small research, which surely include ideas, subjects, technologies, market pressures, etc., which were not explicitly covered in class. In particular, your research needs to look/examine the last 5 years or so, as well as the next 5 years as you can anticipate from available predictions on the subjects.

## 10.2 Technology trends in the past 5 years

For each of the following subjects/matters, describe how technology trends in the past 5 years impacted this subject, in relation to operating systems, (or impacted operating systems in relation to it!). You should be as comprehensive as possible (comment on different relevant aspects, such as performance, cost, etc.)

### 10.2.1 IoT (Internet of Things)

IoT plays a vital part in the development of OS in general. In the past five years, it has been growing significantly, becomes a boot for new innovations in Operating Systems field. There are a great number of IoT OSs that were developed in order to fit this new recent trend such as Contiki-OS, RIOT and Zephyr. They are most well -known for compatibility with hardware restrictions and battery limitation. Consequently, it tackles successfully the resource-constrained problems of IoT gadgets.
In the past, it was a challenge to develop a stable connection to the Internet from an IoT device due to its limitation in term of Hardware. However, most IoT OSs has been developed to have complete networking compatibility despite of its nature constraints (Zikria et al., 2019). Moreover, most of IoT OSs also has supported multi threading, providing a faster compute speed than any Uni-programming supports before.
The overview of common IoT OSs and its abilities is on the table following [1]

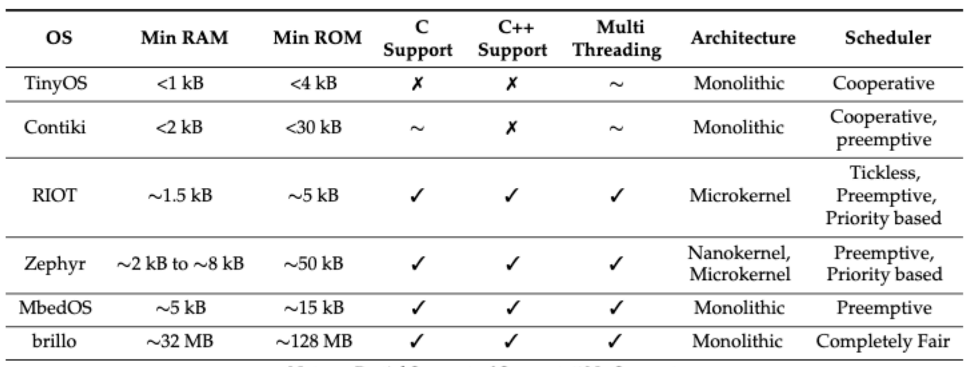


**Table 1.** Overview of IoT OSs.

| OS | Min RAM | Min ROM | C Support | C++ Support | Multi Threading | Architecture | Scheduler |
|---|---|---|---|---|---|---|---|
| TinyOS | <1 kB | <4 kB | ✗ | ✗ | ∼ | Monolithic | Cooperative |
| Contiki | <2 kB | <30 kB | ∼ | ✗ | ∼ | Monolithic | Cooperative, preemptive |
| RIOT | ∼1.5 kB | ∼5 kB | ✓ | ✓ | ✓ | Microkernel | Tickless, Preemptive, Priority based |
| Zephyr | ∼2 kB to ∼8 kB | ∼50 kB | ✓ | ✓ | ✓ | Nanokernel, Microkernel | Preemptive, Priority based |
| MbedOS | ∼5 kB | ∼15 kB | ✓ | ✓ | ✓ | Monolithic | Preemptive |
| brillo | ∼32 MB | ∼128 MB | ✓ | ✓ | ✓ | Monolithic | Completely Fair |

Note: ∼ Partial Support; ✓ Support; ✗ No Support.

Even though the growth of IoT and IoT OSs is considerable, it still exists several issues that need to be address. The problems that engineers are facing can namely be energy efficiency for IoT devices and network connectivity since they are not advanced enough to achieve certain tasks. However, the arises of IoTs not only has greatly inspired the OS field, it also opens a new horizon for OS developers to work on more efficiency systems and overcome the limitations.

### 10.2.2 CPU processing (speed, capacity, cores, etc.)

There is a dramatical changes in CPU processing in recent years. The CPU is now capable of working with Augmented Reality (AR). Thus, it has an impact to the OS to provide the needed API to integrate with the applications. This advancement means the traditional OS with multi threading model need to be improved. In another aspects, there are a number of research on multi core programming to expand the potential of all the cores in CPU nowadays. Multi core programming is the solution for concurrent systems for multi processor systems, and it is being researched carefully across the globe. The field is receiving attention from engineers and computer scientists since it is believed to not only greatly reduce the price point of CPU, but also reduce the dependent on hardware by optimizing the CPU with softwares.
However, the field is currently under developing.

### 10.2.3 Main Memory (i.e. RAM)

Despite the rapid changes in other fields, memory technology is staying essentially the same at its core in the past few years. Although it is noticeable that the main memory now is compact and smaller than ever before with the capacity growing each year, the mechanism to control and manage RAM is still the same. In another word, computer organization remains the same through out the years.

In the other hand, RAM reliability has been greatly improved. The error management now can handle the failure mechanism called *row hammer* in the past. Also, in the past five years, manufacturers have made the price of main memory more accessible to the general public. For example, the price of 8GB RAM in the past 5 years is much higher than nowadays, even with inflation adjusted.

In conclusion, memory technology unlike other technology, has grown stably. It achieved success in extending the limitation of memory compact with a reduced cost, but the integration with Operating Systems seen a little changes.

### 10.2.4 Cloud Computing

In the new decade, many companies and enterprise are changing to cloud computing to reduce the cost for infrastructure, and allow flexibility of the server. This trend effects considerably on how Operating systems is being developing. In the past few years, Internet has been more and more accessible for people all over the world; thus, increases demand for hosting website and management. Internet usage saw a significant increase in recent years. Being inspired by that trend, operating systems are experiencing a rapid development for multiple users instead of the traditional model of one single user, one single system. Nowadays, an user can easily connect to a host and get access to data to process his/her job via the Internet. Cloud computing opens a new perspective for the society as it does not depend solely on the hardware requirements as before. It offers users with great convenience and ease of out dated hardware. With that changes, the OS for cloud computing services have to be developed to accommodate the growing demands. In the past years, the OS to manage the infrastructure of cloud hosting services were dramatically improved to offer more secure, and more accessible to users.

On another aspect, there are some critical issues with Web based OS that companies are facing. Since the database of a business is located in the server of a third party, there is a rising concern about security problems. That requires the OS to evolve and secure its access. Moreover, the OS on the cloud has to be stable since there are many users depend on it for daily basic works.

In conclusion, cloud computing offers a turning point for the development of OS. It shifts the traditional OS to a new aspect of modern work. Despites its existing disadvantages, it is still an evolution of operating systems.

### 10.2.5 Big-data Processing

In recent years, data is proving its important for business. Especially for large enterprises such as Google, Amazon, Facebook,.. Big data is essential to how they gain profits. Therefore, its impact on how operating systems are designed is undoubtedly. For large companies, the OS for their data server must be secured and fast enough for analytics. With these criteria, it can be observed that OS there is a need for OS to develop. Data analytics is also relying on a fast and reliable OS to complete business's work. There were new innovations such as Apache Spark for managing Data Source interface. It provides an easier tool for analyst to achieve the data they need.

## 10.3 Technology trends anticipate in the next 5 years

For each of the following subjects/matters, describe how you anticipate that very recent and future technology trends in the short-term of the next 5 years will impact this subject, in relation to operating systems (or impact operating systems in relation to it). You should be as comprehensive as possible. Remember that your answer should reflect your own anticipation based on your research

### 10.3.1 5G Networks

Nowadays, there is a currently a 5G war between nations in the world. 5G is the fifth generation of wireless communications technologies that is promised to provide the connectivity that was ever available before. With the new technology, the limitation Internet speed might no longer exist in the upcoming years. As a result, cloud computing could be stronger than ever.

On the business side, it will be a challenge that companies have to drive resources into developing Web based OS that is capable to accommodate such huge amounts of requests while securing the database. On the customer aspect, we can expect a shift from traditional OS to Web OS and the rise of Web Applications since it is already very popular and the average Internet usage of people will increase as 5G is starting. More technologies for WebOS could be developed; thus, allowing developers more flexible on either learning or maintain the system. Moreover, unlike the traditional OS, WebOS would allow software providers to maintain and keep track of its user experience much easier. Hence, user's satisfaction can be increased.

In conclusion, 5G technology offers a faster speed to connect to the web. Thus, it empowers the development for new OS innovations.

### 10.3.2 Artificial intelligence

Artificial intelligence (A.I) is the future undoubtedly as more and more businesses are using for a variety purposes such as predict customer behaviours. It is being applied to many fields nowadays as it opens a new aspect of computing and research.

First of all, to train an AI model, the need for computing a large set of data is inevitable. However, personal hardware or computers might not have enough computing speed to execute such a heavy task. Therefore, cloud computing in a remote server is a must for developers working with AI. Operating systems have to evolve to have enough computation power to serve the requests for training AI model.

Furthermore, AI training depends hugely on the data set it is provided. As a result, data set has to be analyzed and sorted in purpose to create an accurate model and produce the correct outcomes. This demands a system of big data efficient enough for big enterprises which rises a demand to implement an OS structure that is reliable and fast enough. Therefore, we could expect a huge development in server OS and system manager in the next few years.

In conclusion, artificial intelligence is a technology that is expanding with an unstoppable speed. It is also rising new OS needs in the future and new technologies might be invented to board these needs.

#### 10.3.3 Main Memory (i.e. RAM)

Memory technology is expected to grow stably in the next 5 years. But it will affect the future of OS development.

Firstly, the memory-centric architectures are gaining attention from researchers. This is because the advancement of processor is expected to reach its peak soon. Moreover, data is becoming the new currency; thus, memory is more valuable than ever before. As this is a rising trend, OS will probably be focused on utilizing the pool of memory and expand its ability for faster memory access. In addition, memory error handling in OS will be improved over time as the risk of losing valuable memory is more dangerous than before. The OS will be evolved such that memory errors at big scale can be minimized.

To be concluded, the next five years might be the years that the world shifts attention to memory-centric OS, and manufacturers start to improve memory technology.

### 10.3.4 Reacting to world pandemics, such as the current COVID-19

In the next five years, the technologies might have been developed enough for the world to cope with another pandemic such as COVID-19 once again.

With the active development of Web OS, and 5G networks, it will be eased for employers to switch workplace from traditional office to home office. The transition might get smoother as people are having better tools. In addition, people will be less depend on hardware systems at home. Since employers can connect to workplace and do usual works regardless of the specs on their home computer, the work flows in an office might remain the same.

Furthermore, with better technologies in terms of connectivity, hosting, computing speed, we can be benefited with more reliable data. With accurate data, nation leaders will have a broader vision on how spread a pandemic are; thus, actions and measures will be taken at the right time, helping to prevent the spread.

With the rate of new development in technologies, it can be predicted that citizens of the world will have a better preparation in the next pandemic.

### 10.3.5 Fault-tolerance

In the future, it is expected that fault will have less impacts than before.

There might be more advanced failure handling to be embedded in the OS to response to the failure. For example, the system might be backed up frequently; therefore, in case of failure, the system can easily shut down and recover the data with the processes it is executing. With a large scale infrastructure, the OS might be implemented so that it triggers a backup system to cope with failure immediately. Thus, in the event of system downs, user experience might not be interrupted. To achieve this goal, the OS in the server has to have a special fast response mechanism for failure. The implementation of such system will be very expensive at the moment. However, with the rapid development of OS, the cost is predicted to be lowed in the future
In conclusion, with the rising of needs for such improvement, most large and

medium infrastructure might be able to afford a system for fault tolerance that is fast enough to not interrupt user experience in the system.