Assignment 5

1.

(a) A process is a program which is in execution.

- **True**

(b) GUI is a part of an Operating System (OS).

- **False**

(c) Implementing thread in user space allows a system to take advantage of multiple processors.

- **False**

(d) In a C programming with multiple threads, the threads always run in sequence.

- **False**

(e) For a problem that can be splitted into multiple sub-problems and the sub-problem can be handled by multiple threads, the more threads are created, the faster the problem is finished.

- **True, when there is substantial computing and also substantial I/O, having threads allows these activities to overlap, thus speeding up the application.**

(f) Multiple kernel space level threads in the same process can make system calls without affecting other threads.

- **True**

(g) Round robin process scheduling minimizes the average response time of a system.

- **False, can do it, but only with perfect quantum-value.**

(h) In the Producer/Consumer problem, we always consume and produce items at the same rate.

- **False**

(i) The first-come first-serve scheduling algorithm maximizes the fairness among processes.

- **False**

(j) RAM is abstracted as a file by an OS.

- **False**

(k) A swapping system eliminates holes by compaction.

- **True**

(l) A page fault occurs when a program attempts to access a virtual page currently in a page frame.

- **False, page fault happends when a program attempts to access a page in the physical memory when it acctually is in the virtual memory**

(m) An unsafe state will always lead to a resource deadlock.

- **False**

2.

(6 points) Describe your understanding about Swapping based on the following questions (within 5 sentences for each)

(a) What is the technical problem that Swapping is supposed to solve?

- When processes ask for too much RAM, swapping and virtual memory is the answer.

(b) How does it work?

- A process is executing in the RAM, the process waits for I/O, and instead of staying in RAM and taking up valuable space, it gets swapped to disk. And

another process can take its place. When the process get I/O, it will swap back again.

(c) What is/are the drawback(s) of Swapping?

- The drawback is that if a process is swapped before its finished, it needs to be swapped back again, and that takes time.
- It also makes holes in memory, to grow (room for growth).

3.

(10 points) Deadlock

(a) Explain shortly (5-7 sentences) what a resource deadlock is.

- Deadlock happens if each process is waiting for an event that only other processes can cause.
- Let's say we have resource R1 and R2, and process 1 and 2.
  - If 1 holds R1, 1 waits for R2, 2 holds R2, 2 waits for R1.
  - As we can see, both processes wait forever.

(b) What are the necessary conditions for a deadlock to occur? Explain each condition in 1-2 sentences.

- Mutual exclusion:
  - At least one resource must be held by a process. Otherwise the processes are not prevented from using the resource
- Hold and wait:
  - a process is currently holding at least one resource and at the same time requesting additional resources which is held by another process.
- Non-preemption:
  - A resource can be released by the process that is holding it
- Circular wait:
  - Each process is waiting for a resource which is held by another process, .. in a loop, which at the end is waiting for the first process to release the resource.

(c) Are the conditions in (b) the sufficient conditions? Explain your answer.

- All for conditions must be present, else no deadlock.

(d) List all approaches that an OS deals with deadlock?

- Preemption, rollback, termination (killing a process)

(e) In the Ostrich algorithm, what does an OS do when there is a deadlock?

- Nothing.

4.

(6 points) List approaches that two or more processes in a machine can use to communicate with each other.

- Semaphore
- Signals
- Sockets
- Pipefiles
- Shared memory
- Queue

5. (10 points) Suppose that a file FileA.dat contains a number of integer numbers. Write a C program that (a) reads all elements of FileA.dat and (b) prints out the summation of the elements in a terminal.

```
int main() {
   int currentNum = 0;
```

```
    int sum = 0;

    File* file = fopen("FileA.dat", "r");
    while(!feof(file)) {
        fscanf(file, "%d", &currentNum);
        sum += currentNum;
    }
    fclose(file);
    printf("Sum: %d", sum);
    return 0;
}
```

6. (10 points) Write a C program that uses multiple threads to print out the message 'Welcome to the Faculty of Computer Science, Østfold University College'. The program has several threads as the followings:
   (a) Thread 1 prints 'Welcome to the '
   (b) Thread 2 prints 'Faculty of Computer Science,'
   (c) The main thread prints 'Østfold University College'
   Use semaphore to synchronize above threads so that the messages they print are always in the right order.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>
#include <semaphore.h>
sem_t semaphore;


void* Thread1PrintMessage(void* ThreadId) {
    printf("Welcome to the ");
    sem_post(&semaphore);
}

void* Thread2PrintMessage(void* ThreadId) {
    sem_wait(&semaphore);
    printf("Faculty of Computer Science, \n");
    sem_post(&semaphore);

}

int main(int argc, char** argv) {
    pthread_t t1;
    pthread_t t2;

    sem_init(&semaphore, 0, 0);
    pthread_create(&t1, NULL, (void *) Thread1PrintMessage,
(void *) 1);
```

```
   pthread_create(&t2, NULL, (void *) Thread2PrintMessage,
(void *) 2);
   pthread_join(t1, NULL);
   pthread_join(t2, NULL);

   printf("Østfold University College");
   sem_destroy(&semaphore);

   return 0;
}
```

7. (10 points) Write a C program that performs matrix multiplication for 2 matrices of size 32x32 using pthreads. MatrixA.dat and MatrixB.dat are the files containing the numbers for the first and the second matrix, respectively. The specifications for the program are as follows:
(a) Using 32 threads each of which is responsible for calculating one row of the output matrix.
(b) If any errors are encountered during program execution, the program prints out the error information and then exits.
(c) The output file is the same format as the input files.

Tested, writes to file. Have not tested if it uses one thread pr. calculation.

```
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>
#include <stdio.h>

// Global variables
int thirtyTwo = 32;
int A[32][32];
int B[32][32];
int C[32][32];

void* matrixMul(void* address) {
   int *x = (int *) address;
   int i = *x;
   for(int j = 0; j < thirtyTwo; j++) {
       for(int k = 0; k < thirtyTwo; k++) {
           // Multiplying matrix A and B to get the right
matrix C.
           C[i][j] += A[i][k] * B[k][j];
       }
   }
}

int main() {
   char *filename[] = {"MatrixA.dat", "MatrixB.dat"};

   for(int x = 0; x < 2; x++) {
       FILE *file = freopen(filename[x], "r", stdin);
```

```
        // Reading the files, and putting them inside the
2D-array.
        for(int i = 0; i < thirtyTwo; i++) {
            for(int j = 0; j < thirtyTwo; j++) {
                if(x == 0) {
                    fscanf(file, "%d", &A[i][j]);
                }
                if(x == 1) {
                    fscanf(file, "%d", &B[i][j]);
                }
            }
        }
        fclose(file);
    }

    // Creating thread
    pthread_t t[thirtyTwo];
    int send[thirtyTwo];
    for(int i = 0; i < thirtyTwo; i++) {
        send[i] = i;
        int err = pthread_create(&t[i], NULL, matrixMul,
(void*)&send[i]);
        if(err != 0) {
            perror("Error");
            exit(-1);
        }
    }

    for(int i = 0; i < thirtyTwo; i++) {
        pthread_join(t[i], NULL);
    }

    // Writeing to matrix C to MatrixC.txt
    FILE *file = fopen("MatrixC.dat", "wb");
    for(int i = 0; i < thirtyTwo; i++) {
        for(int j = 0; j < thirtyTwo; j++) {
            fprintf(file, "%d ", C[i][j]);
        }
        fprintf(file, "\n");
    }
    fclose(file);
    printf("Result is written to MatrixC.dat!\n");

    return 0;
}
```

8. (5 points) Consider a swapping system in which the memory consists of the following hole sizes in memory order: 10MB, 4MB, 20MB, 18MB, 7MB, 9MB, 12MB and 15MB. Suppose that the Best-Fit algorithm is used to allocate memory for a new process, which hole is taken for successive segment requests of
   (a) 12 MB - 12MB
   (b) 10 MB - 10MB
   (c) 9 MB - 9MB

Holes remaining: 4MB, 20MB, 18MB, 7MB, 15MB

9. (5 points) A virtual memory is divided into 4-KB pages. For each of the following decimal virtual addresses, compute the corresponding virtual page number and offset
   (a) 4096 - Page 2, offset 0
   (b) 8200 - Page 3, offset 8
   (c) 65540 - Page 17, offset 4
   Used:
   - Offset = x % 4096
   - Page = (x / 4096) + 1 without decimals

10. (10 points) Consider a system which has three processes P1, P2, and P3 and four types of resource A, B, C, and D. Each resource has exactly 2 available instances. To complete, each process needs one of each resource. For each of the following scenarios, indicate whether the state is deadlock or not. If the state is not deadlock, provide the sequence of process execution. If the state is deadlock, recommend a way to recover from deadlock.
    (a) Scenario 1: P1 has 1 B, 1 C, and 1 D; P2 has 1 A, and 1 C; P3 has 1 A and 1 D.
    -

	A	B	C	D
P1	0	1	1	1
P2	1	0	1	0
P3	1	0	0	1

    - Deadlock
    - Can be recovered with preemption → Take A from P3 and give it to P1 → When P1 is finished, P3 get A back.

    (b) Scenario 2: P1 has 1 A and 1 D; P2 has 1 B and 1 C; P3 has 1 A and 1 C.
    -

	A	B	C	D
P1	1	0	0	1
P2	0	1	1	0
P3	1	0	1	0

    - Start with P3, then P2 / P1 (doesn't matter at this point). Can run at the same time?

11. (10 points) Consider a system which has four processes A, B, C and D and five types of resources. The current allocation matrix and requested matrix for the processes are as the following, respectively

Current allocation

	R1	R2	R3	R4	R5
A	1	2	0	2	1
B	1	0	1	1	0
C	1	1	1	0	1
D	0	0	1	0	1

Requested allocation

	R1	R2	R3	R4	R5
A	0	0	1	1	1
B	0	1	0	1	0
C	1	3	0	1	0
D	0	1	2	1	0

Suppose that the Available resource is Av = (0 1 x 1 0).

(a) What is the smallest safe value of x? for which this is a safe state? Explain your answer.

Av = (0 1 x 1 0) | temp x = 0

B runs and terminates → Av = (1 1 1+x 2 0) | x = 0, increase x cuz D need 1 more R3

D runs and terminates → Av = (1 1 2+x 2 1) | x = 1

A runs and terminates → Av = (2 3 2+x 4 2) | x = 1

C runs and terminates → Av = (3 4 3+x 4 3) | x = 1

Smallest safe value of x = 1.

(b) How many of each resource exists on the system?

Resources that exists = (3 4 3+ 4 3)

12. (5 points) Suppose that you have to build an OS for an internet-of-thing (IoT) device. The features of the IoT device are as follows:

(a) It has a limited memory.

(b) The device collects data and sends the data to a base station (BS).

(c) As soon as the device has data, it must send the data immediately.

What would be the requirements for the OS that can be built for this IoT device?

This is not the correct solution!!!

- CPU - One core
- RAM - 500MB
- Disk - 5GB

- Wireless connection