Midterm Test   
CSCE 313 Summer 2020  
120 points

Date: June 10, 2020

Student Name: Joshua Chong

UIN: 227009490

I have adhered to Aggie Code of Honor

Signature: Joshua Chong

TRUE/FALSE Questions [1 pts each]

1. In multiprogramming, multiple jobs are preloaded in memory to save time switching them

True

1. Interrupts are mandatory to bring “paused” programs back to CPU.

True.

1. Multiprogramming is ineffective when all running programs are made of “CPU-only” operations

True, because Multiprogramming allows I/O and CPU to be used at the same time, if it’s only CPU, then it may as well be the same as sequential.

1. Interrupts are checked between every two CPU instructions

True

1. **Efficiency** is the secondary goal of an OS, right after the primary goal of **Convenience**

True, as strange as it may sound.

1. During every context switch, the scheduler runs to decide which process to run next

True, as the scheduler is the algorithm that determines the next process like round robin or shortest time left.

1. Context switch can cause significant overhead if implemented inefficiently

True, if the context switch is significant like the 0.5 second example from the review, it will be more trouble than it’s worth.

1. Every process gets its own private address space that is isolated from other processes

True, I remember reading that the process doesn’t need to know or conflict with other processes, therefore it must be isolated to operate in such a manner where the OS serves as the middle man.

1. Kernel is mapped to every process’s address space

True, by this there is space for the kernel in every process’s address space that the process can’t touch.

1. Memory protection (to protect kernel) is implemented in hardware

True, it’s done so with power above the kernel (being hardware) such that miscellaneous processes cannot access memory outside their own domain and not affect others either.

1. Opening a file is an example of Asynchronous Interrupt

True.

1. A **Trap** always return control to the immediately following instruction.

False.

1. Implementation of a System Call requires context switch and mode elevation

True.

1. A Fault returns control (if it does at all) to the same instruction

True. (It’s the CPU that returns control after the fault is successfully handled but I’m assuming that it’s indirectly highlighting the “same instruction” portion to which it does return to the same and not the following instruction.)

1. A page fault may lead to a segmentation fault

True

1. A Round Robin (RR) scheduler is always better than FIFO

False

1. SRTF guarantees shortest Average Response Time (ART)

False

1. FIFO is a fair scheduling method

False. It takes each in as they come and until they are completed.

1. Increasing higher device utilization increases the system throughput

True.

1. Larger RR time quantum may lead to unresponsive behavior for some I/O bound tasks (e.g., IPads touch events, mouse clicking)

False, context switch happens once a job is complete, meaning there’s not a gap in between tasks if one job is completed early on in the time quantum.

Short Questions (be precise, long answers are not expected/desired)

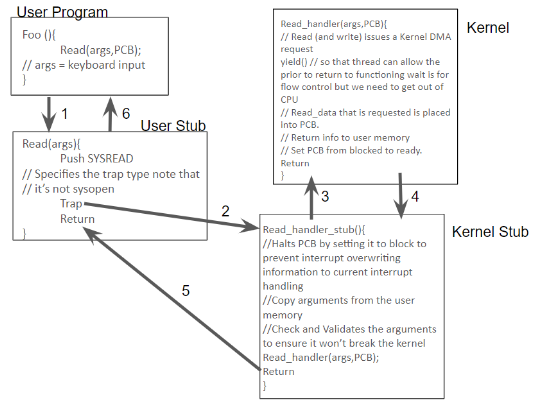
1. [10 pts] Describe how I/O operation by 1 program can happen simultaneously with CPU operations by other program(s). What is the main technique and what hardware component(s) is/are involved in this?

The main technique is multiprogramming that is able to overlap both I/O and CPU operations. The Direct Memory Access controller prevents the CPU doing a job to clash with the I/O hardware/device. The programs must be preloaded on the memory with the monitor such that it can be accessed by the CPU for increased utilization (Of the CPU).

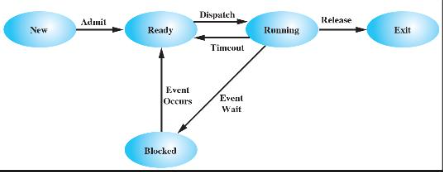
1. [5 pts] Can a multi-programmed system work without Interrupts? Why or why not?

If there’s no interrupts, the DMA (When an action is completed) couldn’t notify the CPU it’s done. Without the communication of the DMA to the CPU, the I/O and CPU jobs would clash. Therefore, the DMA would be useless without interrupts and it’s already known that for multiprogramming to work, the DMA must prevent the I/O interference with the CPU.

1. [15 pts] Describe the steps necessary to perform the read() system call for getting a 100-byte record from a disk file. What different states the process (performing the read operation) can be in? Discuss using the process state diagram.



Different states the process could be in would be Ready, Running, and Blocked.



1. [15 pts] Assuming each page is 4KB, how many page faults the following program will generate? Can the number of faults be different from one system to another? Explain your answer.

class Header{

public:

    int data;

    bool status;

};

int main (){

    int size = 4 \* 1024 \* 1024; // 4 mega bytes

    char \* memory = new char [size]; // allocate memory

    Header\* h = (Header\*) memory;

    Header ho;

    // copy header object ho onto memory count times

    int count = 512 \* 1024;

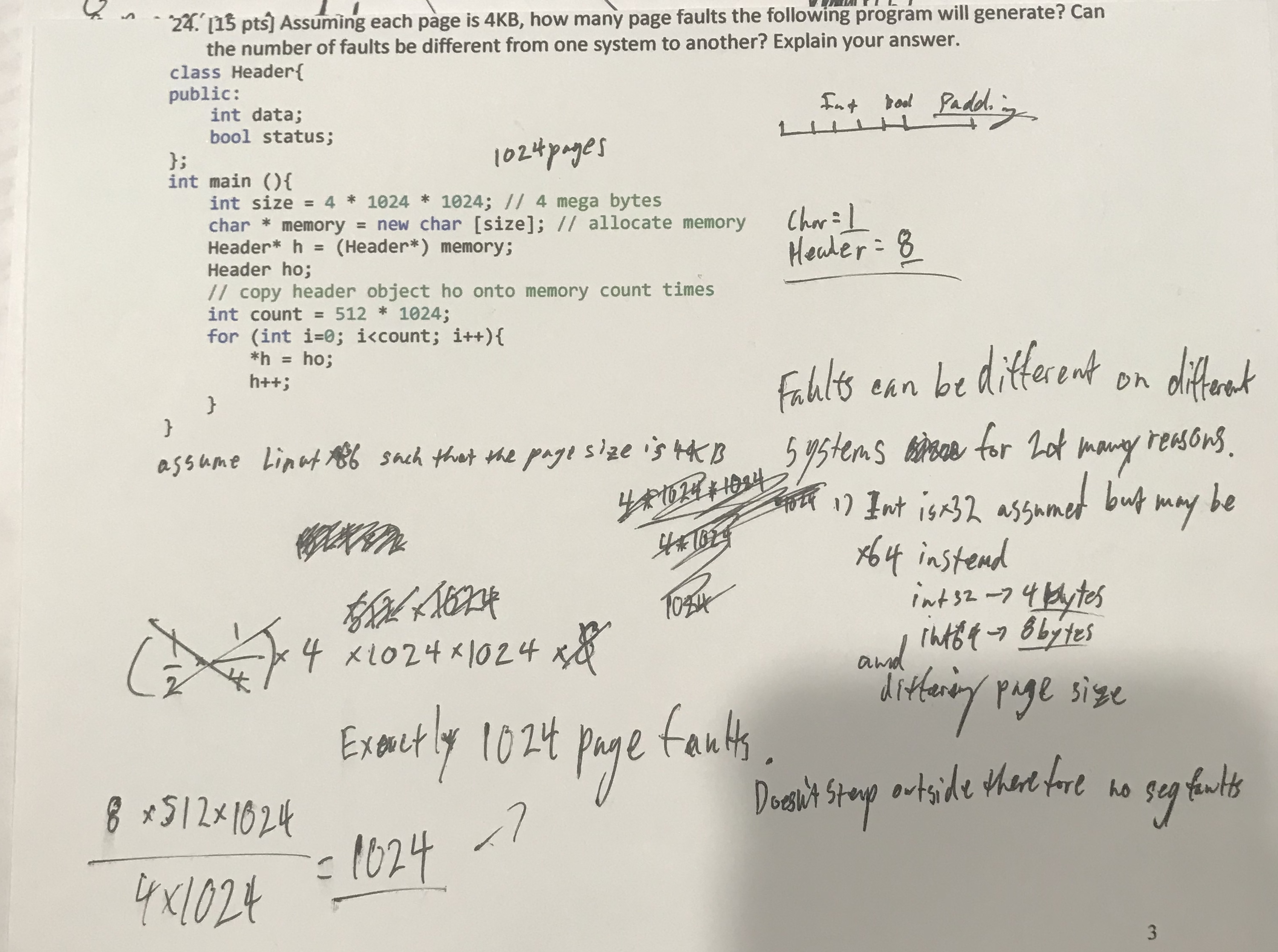
    for (int i=0; i<count; i++){

        \*h = ho;

        h++;

    }

}



Noted that there’s padding in the header of 3 bytes.

1. [20 pts] Consider the following program. Assume that process IDs are always increment-by-1 and the number depends on which order they are created. For instance, if process X is created after process Y, the PID of process X is greater than that of Y. Now, what will be the output of the following program? How many processes will be created? Describe with help of a process tree and based on how wait() function works.

You can assume that the first process’s ID is 1000 and add 1 for each subsequent process in the order they are created

**for (int i=0; i<3; i++){**

**int f = fork();**

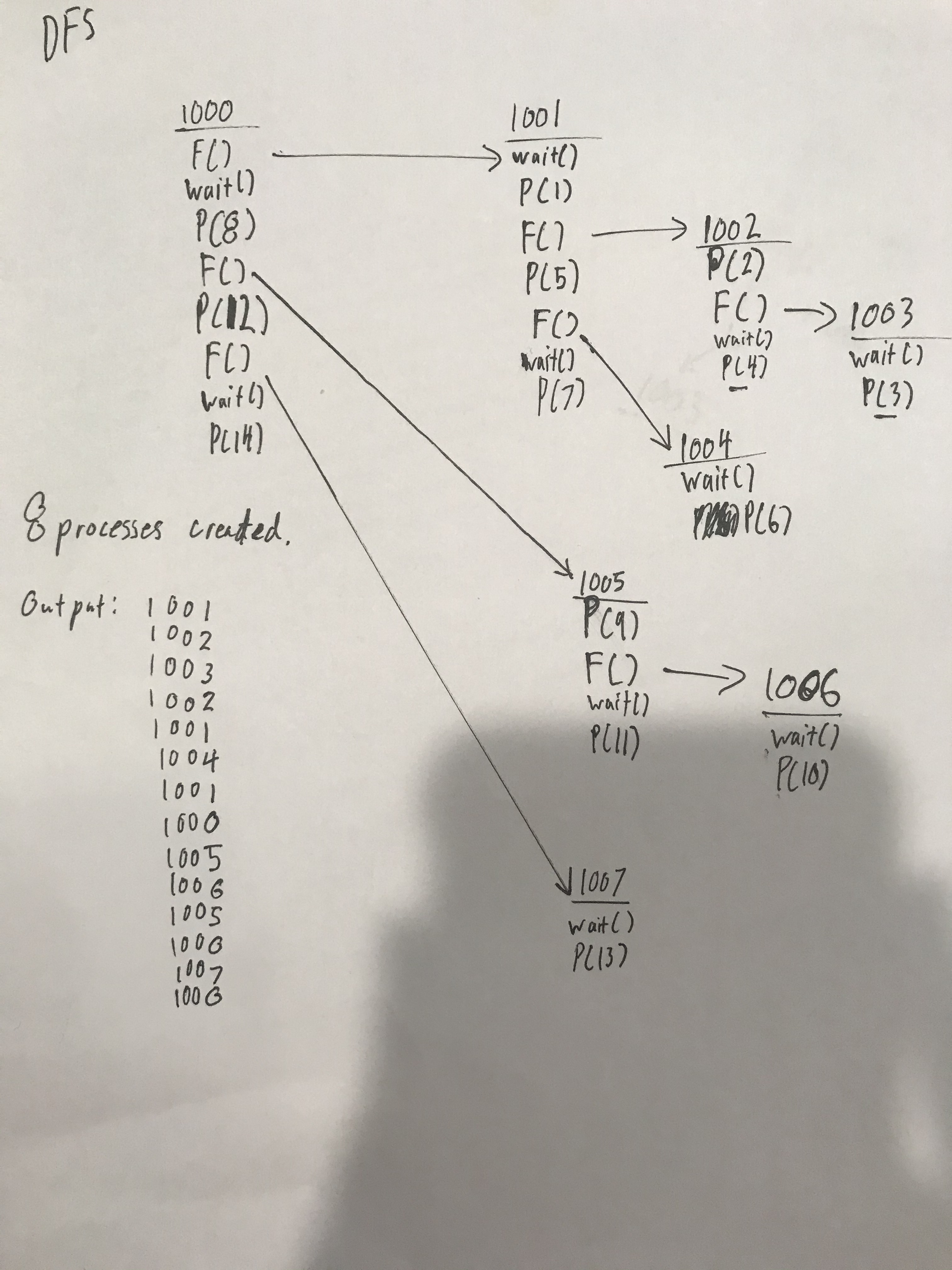
**if (i == 0 || i == 2)**

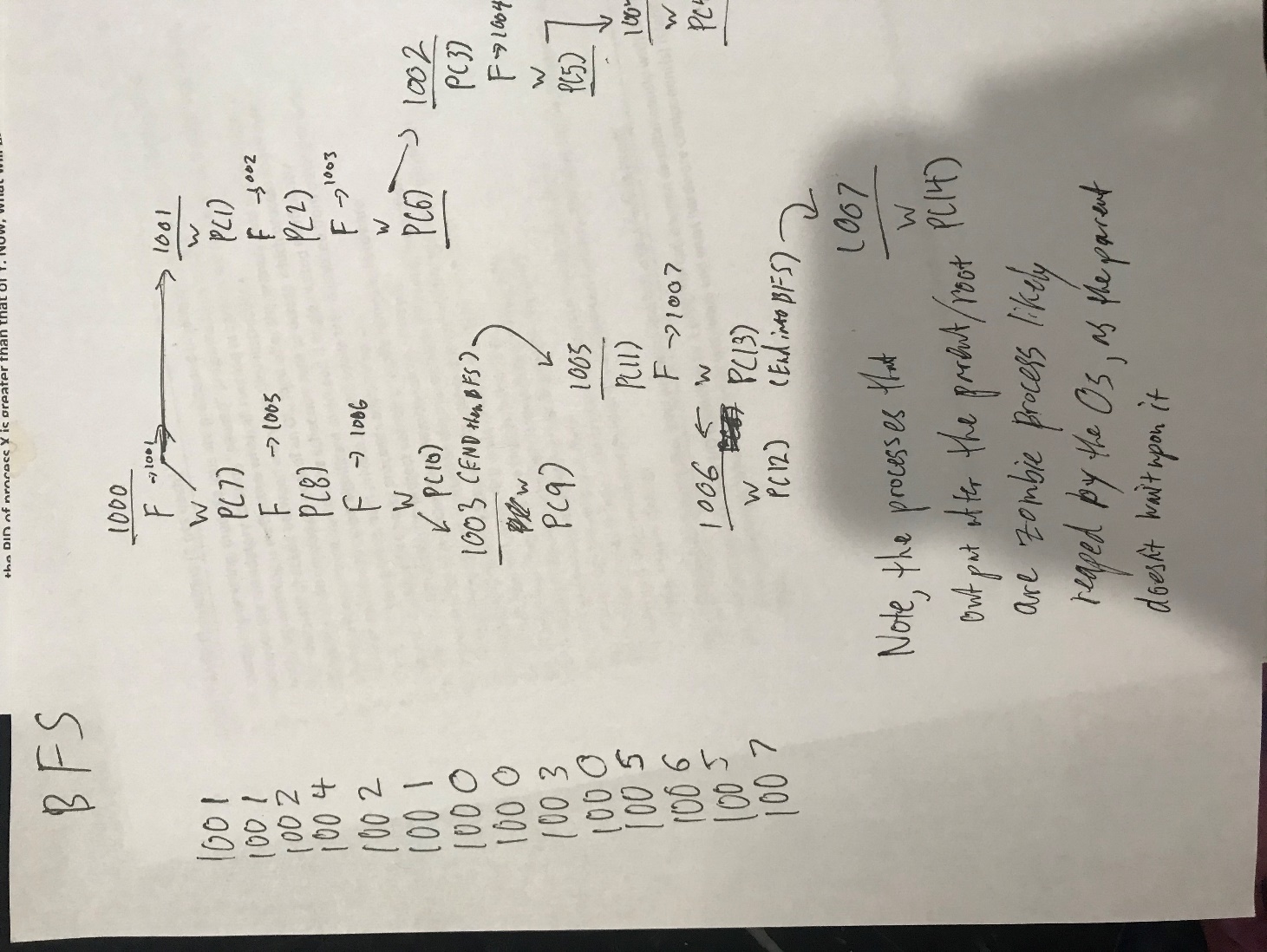
**wait(0);**

**cout << "PID = "<< getpid() << endl;**

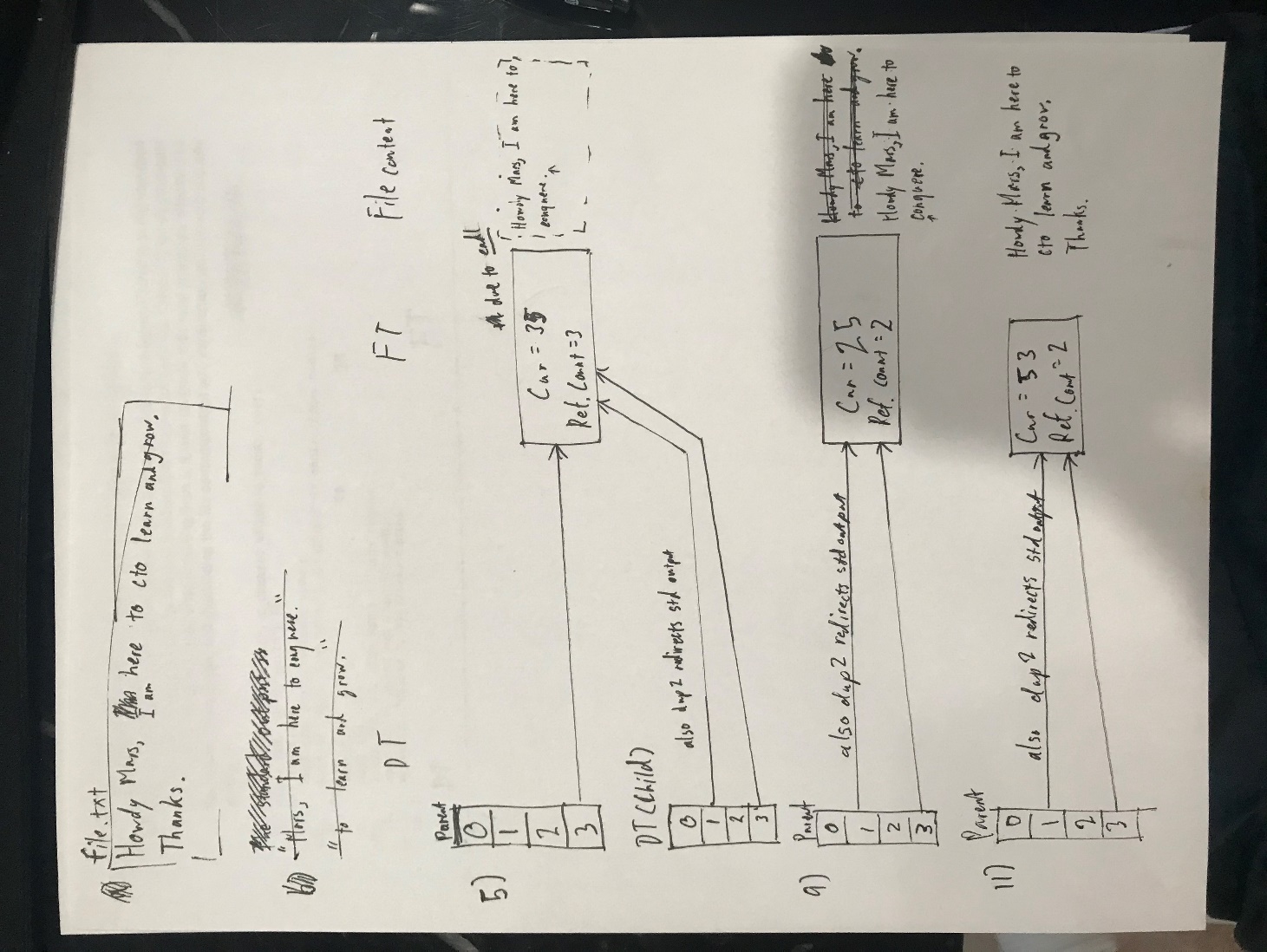
**}**

Wait will reap any arbitrary child processes when there’s more than one child process available to reap, this will cause issues in BFS as there’s many solutions depending on which child process it chooses to reap in the inconsistent order.





1. [20 pts] What will the file file.txt contain after running the following program? What is in the standard output? Explain your answer by drawing the Descriptor Tables (DT) and File Table (FT) for all processes (i.e., from this program) right after executing lines 5, 9, and 11. (Note that most points are allocated to the correct explanation. Just mentioning the file content/output w/o explanation will not earn you any point).
2. int fd = open("file.txt", O\_CREAT|O\_WRONLY|O\_TRUNC, 644);
3. write (fd, "Howdy world", 6);
4. if (!fork()){
5. dup2 (fd, 1); //redirect stdout
6. cout << "Mars, I am here to conquere." << endl; //draw tables
7. }else{
8. wait(0);
9. dup2 (fd, 1);
10. lseek (fd, 25, SEEK\_SET); // draw tables
11. cout << "to learn and grow."<< endl;
12. write(fd, "Thanks.", 7);//draw tables
13. }



There’s also supposed to be an arrow right at the period after thanks for line 11 to indicate the Cur placement as seen with the other ones.

23. [15 pts] Assuming 0.5 sec overhead per context switch (i.e., time between 2 user processes) in a 1-CPU-1-core system, schedule the following workload and compute Average Response Time (ART) under:  
a. Shortest Remaining Time First (SRTF) (preemptive)  
b. Round-Robin (RR) with time quantum=2sec

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time(s) | CPU Service Time(s) |
| P1 | 1 | 3 |
| P2 | 3 | 4 |
| P3 | 2 | 2 |
| P4 | 2 | 1 |
| P5 | 4 | 3 |

