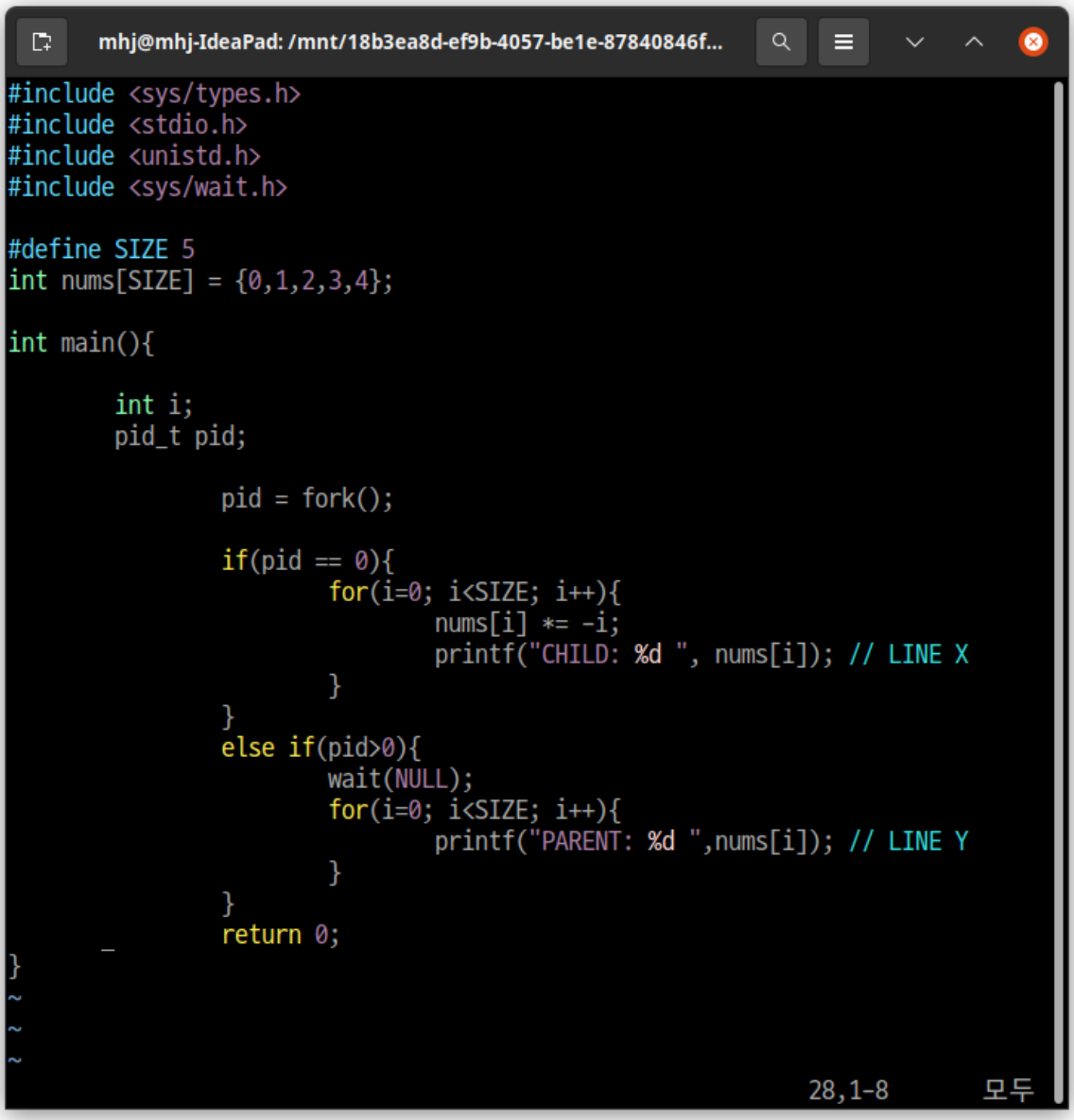
Assignment#1

202033762 장민호

Operating System

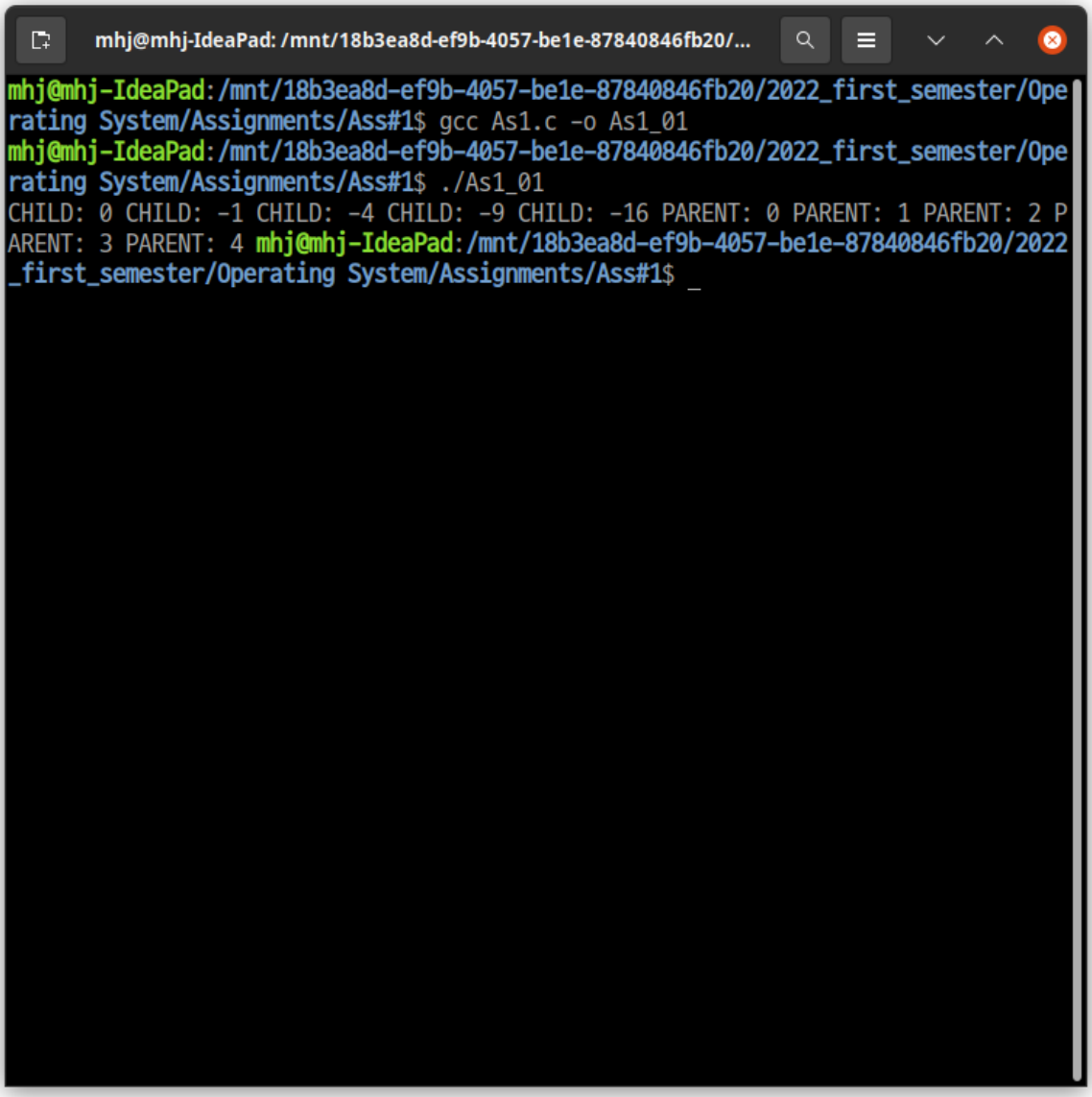
Write your C codes using the VI(M) editor in Linux OS. Compile and execute using the command line. Submit the following:

- **Screen capture of your final codes using VI(M) editor.**




```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

#define SIZE 5
int nums[SIZE] = {0,1,2,3,4};

int main(){

        int i;
        pid_t pid;

                pid = fork();

                if(pid == 0){
                        for(i=0; i<SIZE; i++){
                                nums[i] *= -i;
                                printf("CHILD: %d ", nums[i]); // LINE X
                        }
                }
                else if(pid>0){
                        wait(NULL);
                        for(i=0; i<SIZE; i++){
                                printf("PARENT: %d ",nums[i]); // LINE Y
                        }
                }
                return 0;
}
~
~
~
```

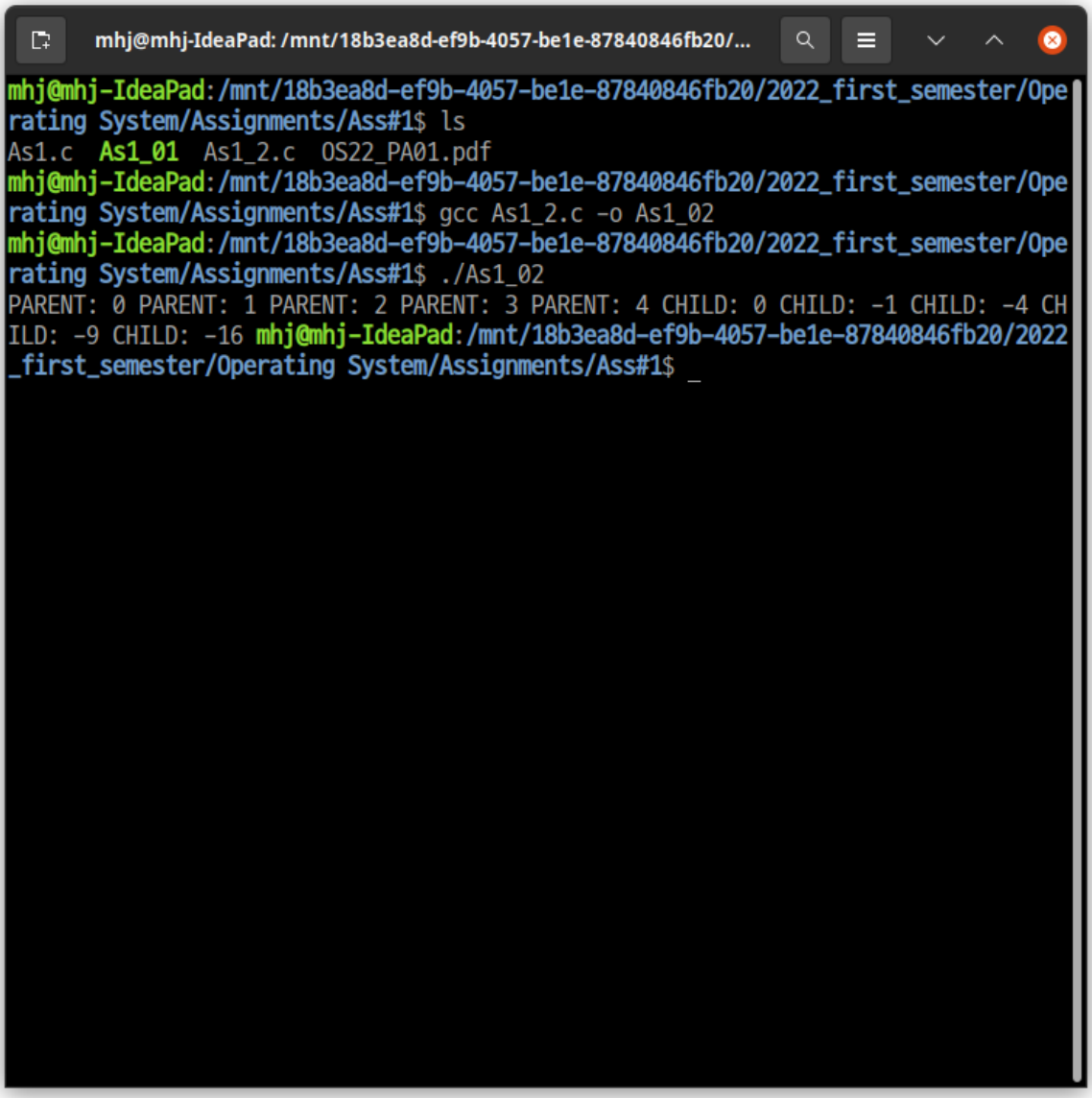
- **Screen capture of compilation, execution and command line results.**



- **Also explain why these results happened.**

Due to WAIT (NULL), the parent process is in a state waiting for the child process to end. Therefore, LINE Y is executed after LINE X is terminated. Therefore, LINE X shows the following results. `CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16` At this time, the values of the numbers in nums[] were changed in LINE X, but since the fork() function creates an independent space between parents and child, the changes in nums[] in child do not affect parents. Therefore, LINE Y shows the following results. `PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4'

- **Now, delete the line “wait(NULL);” and recompile and run. Again, screen capture compilation, execution and command line results.**



- **Explain why the second results are different from the first results.**

When WAIT (NULL) is gone, there is no reason for the parent process to wait until the child process is terminated. Then, the CPU scheduler determines the question of who runs first, parent or child. Looking at the results, it can be seen that the parent process has terminated first. Therefore, the result of LINE Y is `PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4` , followed by the child process terminated, and the result of LINE X is `CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16`.