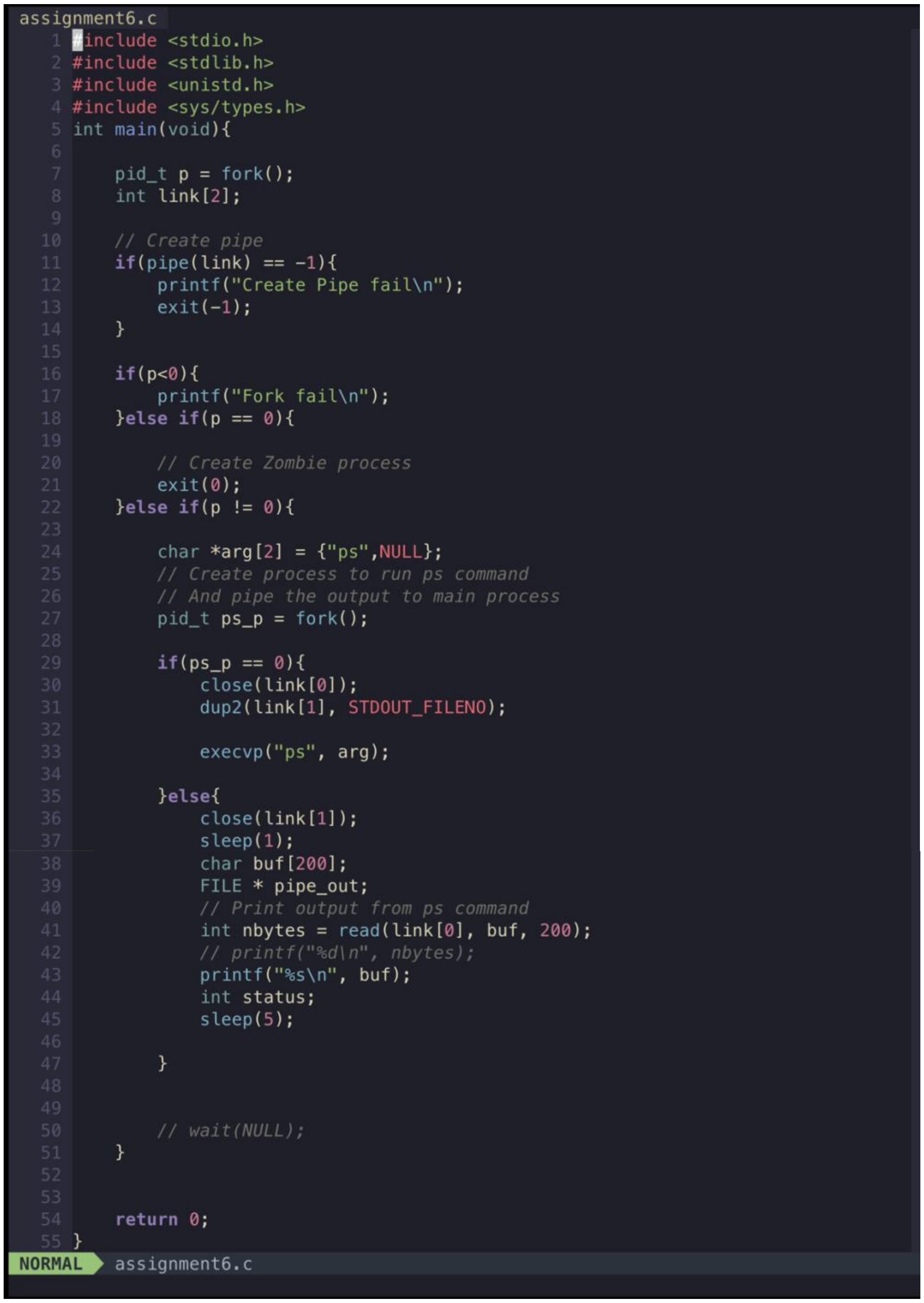
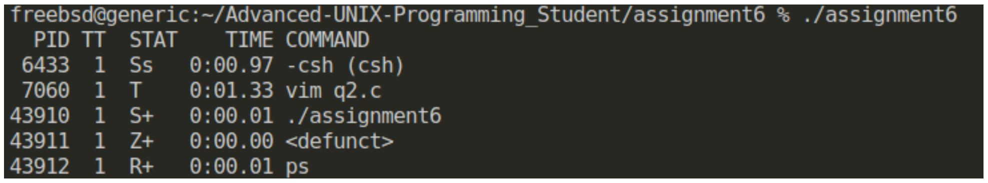
CS5243 Advanced UNIX Programming
Assignment 6(4 pts)
Group 4

Screenshot of codes:



```
assignment6.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
int main(void){

    pid_t p = fork();
    int link[2];

    // Create pipe
    if(pipe(link) == -1){
        printf("Create Pipe fail\n");
        exit(-1);
    }

    if(p<0){
        printf("Fork fail\n");
    }else if(p == 0){

        // Create Zombie process
        exit(0);
    }else if(p != 0){

        char *arg[2] = {"ps",NULL};
        // Create process to run ps command
        // And pipe the output to main process
        pid_t ps_p = fork();

        if(ps_p == 0){
            close(link[0]);
            dup2(link[1], STDOUT_FILENO);

            execvp("ps", arg);

        }else{
            close(link[1]);
            sleep(1);
            char buf[200];
            FILE * pipe_out;
            // Print output from ps command
            int nbytes = read(link[0], buf, 200);
            // printf("%d\n", nbytes);
            printf("%s\n", buf);
            int status;
            sleep(5);

        }


        // wait(NULL);
    }


    return 0;
}
NORMAL  assignment6.c
```

Screenshot of result:



```
freebsd@generic:~/Advanced-UNIX-Programming_Student/assignment6 % ./assignment6
  PID TT  STAT    TIME COMMAND
 6433  1  Ss   0:00.97 -csh (csh)
 7060  1  T    0:01.33 vim q2.c
43910  1  S+   0:00.01 ./assignment6
43911  1  Z+   0:00.00 <defunct>
43912  1  R+   0:00.01 ps
```

1. We use pipe to communicate between processes. We fork child process and exit(0) immediately to make it zombie.
2. Then we use execvp to called UNIX ps(1), and called sleep(1) for the child to sleep long enough to terminate it.
3. In the result, STAT Z+ and <defunct> show that the process is a zombie process.