CS342 Project 1
Berfin KÜÇÜK
21502396

The experiment is done with producer ./producer 10000 | ./consumer 10000 command. First of all producer.c and consumer.c is shown below. And at then, results and interpretations are given. Finally, some part of bilshell.c is included at the end to show functionality of pipe communication.

producer.c : character array with size 10000 has been created and written with printf command

```
#include<fcntl.h>
#include<sys/time.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<string.h>


#define SIZE 10000

int main(){
    char theArray[SIZE];
    int k;
    for( k = 0; k < SIZE; k++){
        theArray[k] = 'b';

    }

    //write
    printf(theArray);
}
```

consumer.c : character array with size 10000 has been read with read syscall

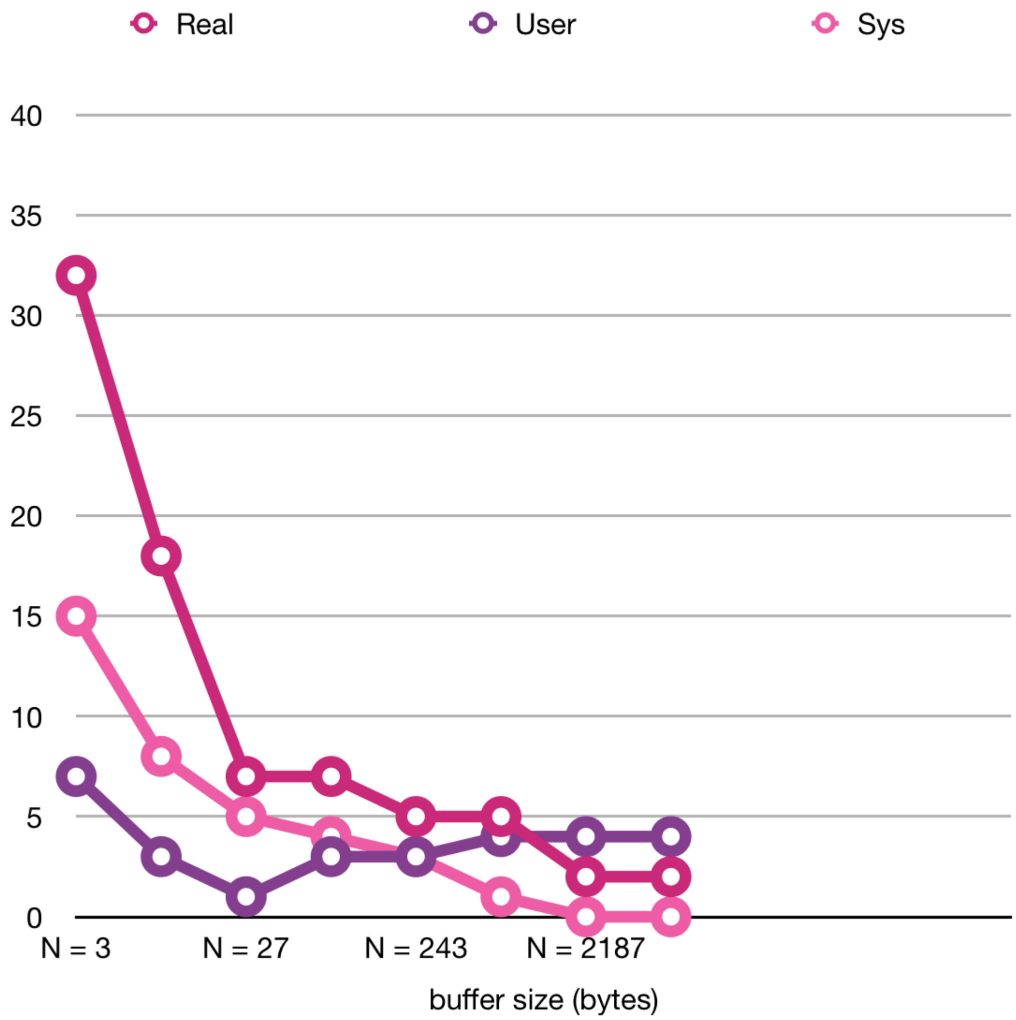
```
#include <stdio.h>
#include <stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/time.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<string.h>

#define SIZE 10000
#define READ_END 0
#define WRITE_END 1
int main()
{
    char readmsg[SIZE];
    int thefd[2]; //pipe

    pipe(thefd); //creating a pipe
    close(thefd[WRITE_END]); //close write end

    read(thefd[0],readmsg,SIZE);

    return 0;
}
```



Graph: buffer size vs ms

As seen in the graph, as N increases, the time spend on kernel mode is dramatically decreased because number of syscall decrease as N increase. However, as seen, after a while there is no significant decrease as it gets bigger.

N is logarithmic as power of 3
M is 10000 as stated above

character-count 10000 (always)
read-call-count 5000 2500 1250 625 313 157 79 respectively as N gets value from 3 to 2187

```
void pipeExec(char **first, char **second){
    int fdpipe1[2]; //file desc. pipe1 of the first child
    int fdpipe2[2]; //fd pipe2 of the second child
    pid_t pid1,pid2; //child processes
    char read_msg[N];

    //creating pipes
    if(pipe(fdpipe1) == -1 ){fprintf(stderr,"Pipe1 Failed"); }

    if(pipe(fdpipe2) == -1 ){fprintf(stderr,"Pipe2 Failed"); }

    //creating child1 and executing first command
    pid1 = fork();
    if(pid1 < 0){fprintf(stderr,"Fork Failed(Child-1)"); }

    //write end is open while read and is closed?
    if (pid1 == 0){
    dup2(fdpipe1[1],1);

    close(fdpipe1[READ_END]);


        execvp(first[0],first);

    //close(fdpipe1[WRITE_END]);                    //SEE HERE
    }
    else { //parent process
    //wait(NULL);
    close(fdpipe1[WRITE_END]); //close write end (index 1) - it is unused here


    int n = 0;
    int m = 0;
    while( n = read(fdpipe1[READ_END], read_msg, N)){
        m = m + n;
        write(fdpipe2[WRITE_END], read_msg ,(int) strlen(read_msg) + 1);
    }

    printf("Number of characters read %d",m);

    //int m = write(fdpipe2[WRITE_END], read_msg ,(int) strlen(read_msg) + 1);
    //printf("Number of characters wrote %d",m);


    //create second child
    pid2 = fork();
    if(pid2 == 0){ //second child process
        dup2(fdpipe2[0],0);
        close(fdpipe2[WRITE_END]);

        execvp(second[0],second);
    }


}}
```