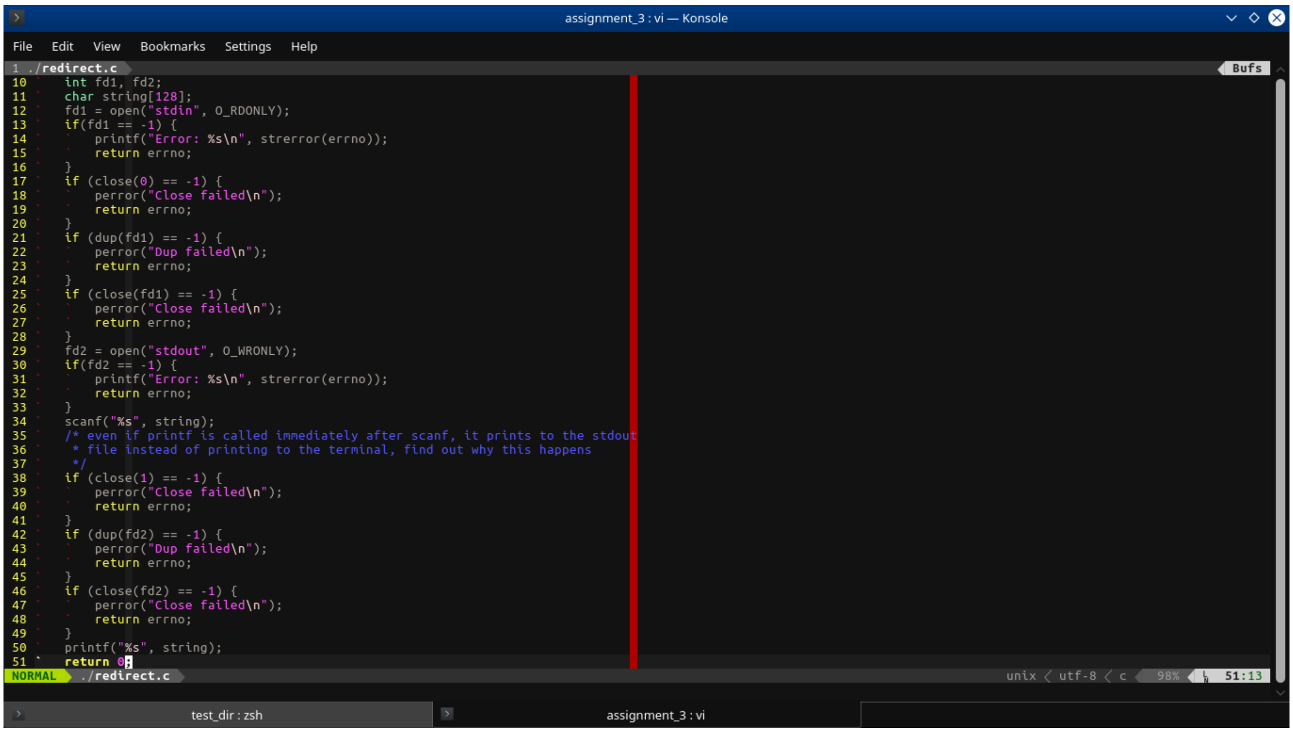
Assignment 3

Srishti Shelke 111603056
Niramay Vaidya 111605075

1. Using dup function redirect stdin to file1 and stdout to file2. Read a line using scanf and write the same using printf. Verify the contents of both files.

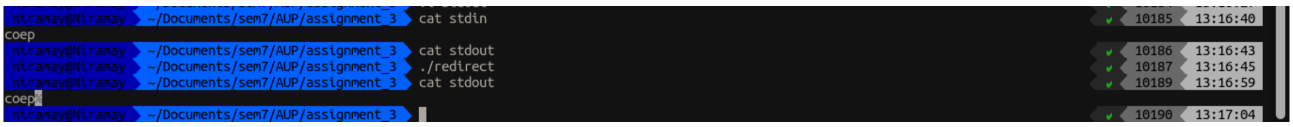
Code-

(Only a single word demonstration has been done using scanf and printf, since scanf only scans up to a whitespace character, since this is enough to demonstrate the file descriptor redirect functionality using dup)



```c
	int fd1, fd2;
	char string[128];
	fd1 = open("stdin", O_RDONLY);
	if(fd1 == -1) {
		printf("Error: %s\n", strerror(errno));
		return errno;
	}
	if (close(0) == -1) {
		perror("Close failed\n");
		return errno;
	}
	if (dup(fd1) == -1) {
		perror("Dup failed\n");
		return errno;
	}
	if (close(fd1) == -1) {
		perror("Close failed\n");
		return errno;
	}
	fd2 = open("stdout", O_WRONLY);
	if(fd2 == -1) {
		printf("Error: %s\n", strerror(errno));
		return errno;
	}
	scanf("%s", string);
	/* even if printf is called immediately after scanf, it prints to the stdout
	 * file instead of printing to the terminal, find out why this happens
	 */
	if (close(1) == -1) {
		perror("Close failed\n");
		return errno;
	}
	if (dup(fd2) == -1) {
		perror("Dup failed\n");
		return errno;
	}
	if (close(fd2) == -1) {
		perror("Close failed\n");
		return errno;
	}
	printf("%s", string);
	return 0;
```

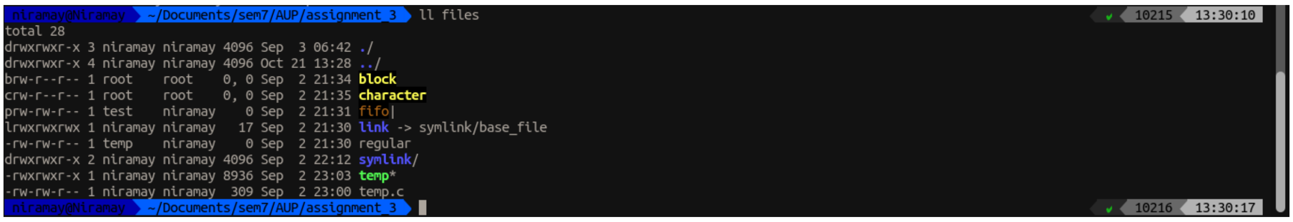
Output-



```
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  cat stdin                     ✔  10185  13:16:40
coep
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  cat stdout                    ✔  10186  13:16:43
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  ./redirect                    ✔  10187  13:16:45
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  cat stdout                    ✔  10189  13:16:59
coep%
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3                                ✔  10190  13:17:04
```

3. Write a program that prints the owner and file type of files. By inputting a directory, the program should read the directory and print the above information for all files in the directory.

The directory content of files directory shown in the actual submission is as below-



```
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  ll files
total 28
drwxrwxr-x 3 niramay niramay 4096 Sep  3 06:42 ./
drwxrwxr-x 4 niramay niramay 4096 Oct 21 13:28 ../
brw-r--r-- 1 root    root    0, 0 Sep  2 21:34 block
crw-r--r-- 1 root    root    0, 0 Sep  2 21:35 character
prw-rw-r-- 1 test    niramay    0 Sep  2 21:31 fifo|
lrwxrwxrwx 1 niramay niramay   17 Sep  2 21:30 link -> symlink/base_file
-rw-rw-r-- 1 temp    niramay    0 Sep  2 21:30 regular
drwxrwxr-x 2 niramay niramay 4096 Sep  2 22:12 symlink/
-rwxrwxr-x 1 niramay niramay 8936 Sep  2 23:03 temp*
-rw-rw-r-- 1 niramay niramay  309 Sep  2 23:00 temp.c
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3 
```

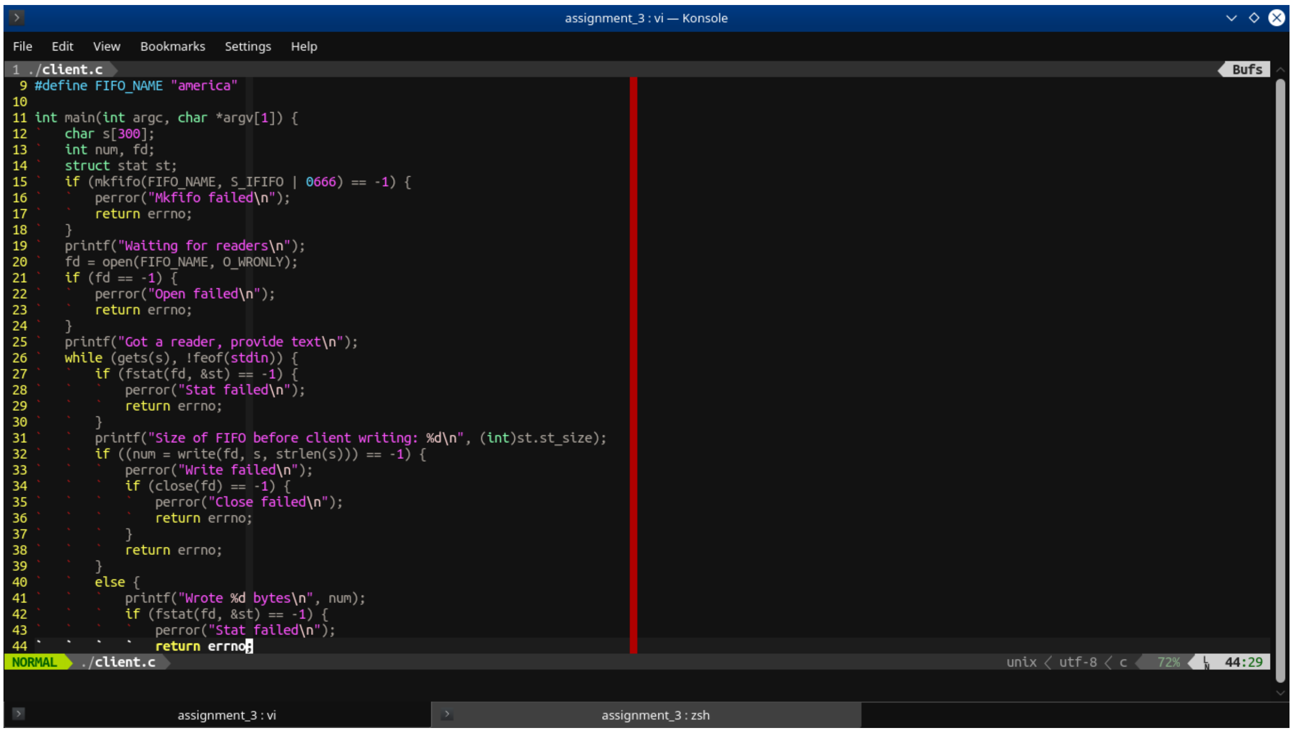
4. Create a FIFO file and write the programs for client-server communication. Print the size of the FIFO file during:

1. Before client-server starts writing to FIFO
2. After client writing a message, but before the server reading it.
3. After client writing a message and after the server reading it.

Describe your observation and understanding

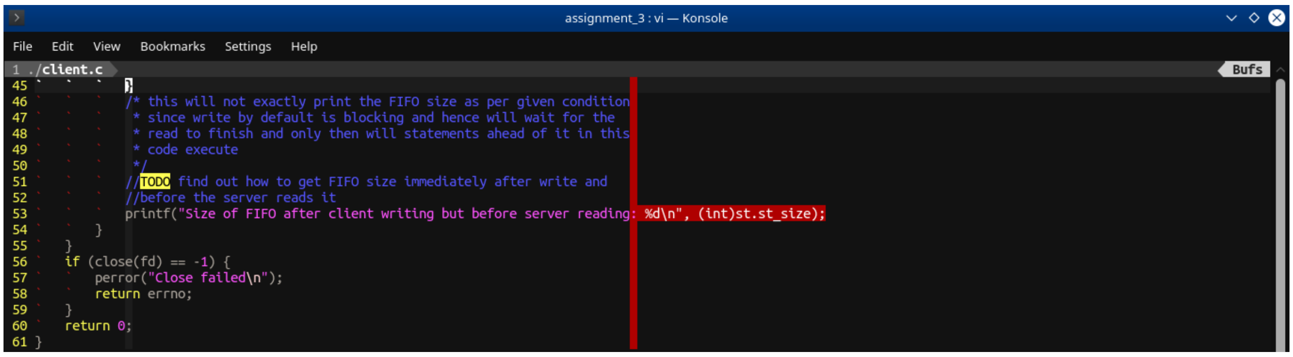
Code-

Client-



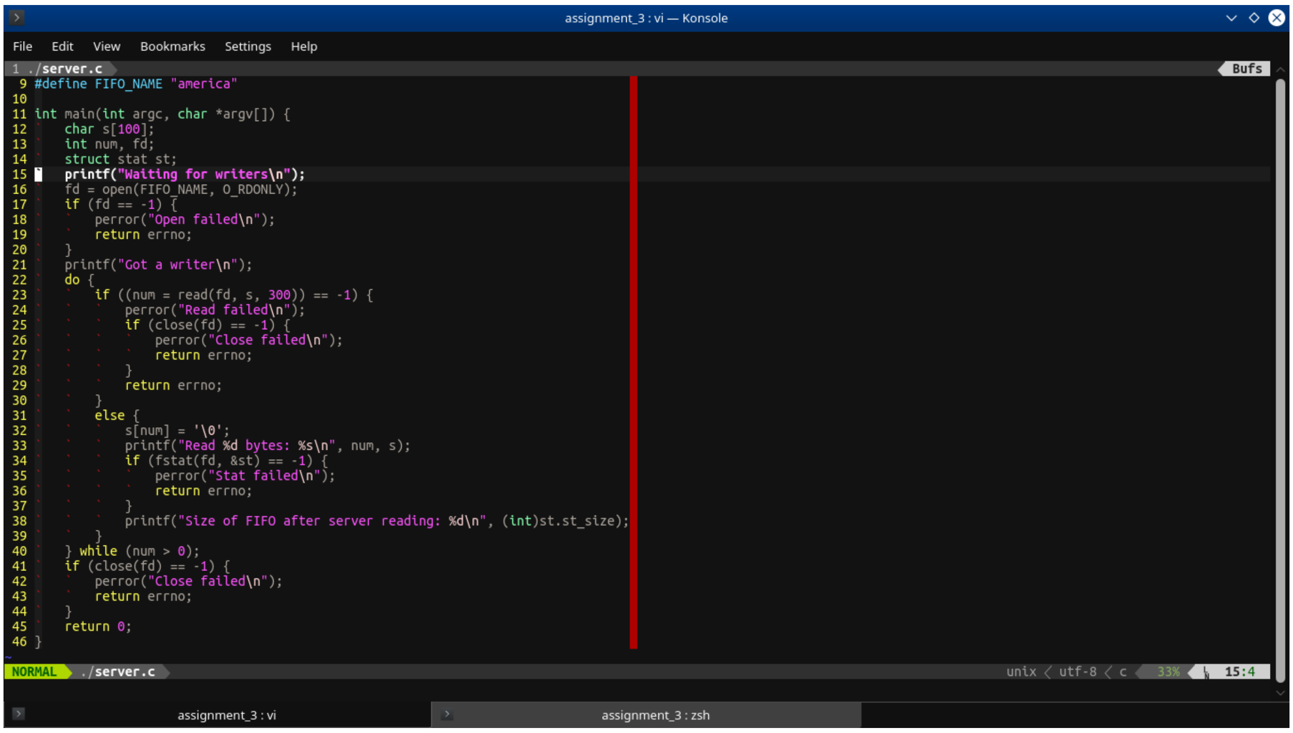
```c
#define FIFO_NAME "america"

int main(int argc, char *argv[1]) {
	char s[300];
	int num, fd;
	struct stat st;
	if (mkfifo(FIFO_NAME, S_IFIFO | 0666) == -1) {
		perror("Mkfifo failed\n");
		return errno;
	}
	printf("Waiting for readers\n");
	fd = open(FIFO_NAME, O_WRONLY);
	if (fd == -1) {
		perror("Open failed\n");
		return errno;
	}
	printf("Got a reader, provide text\n");
	while (gets(s), !feof(stdin)) {
		if (fstat(fd, &st) == -1) {
			perror("Stat failed\n");
			return errno;
		}
		printf("Size of FIFO before client writing: %d\n", (int)st.st_size);
		if ((num = write(fd, s, strlen(s))) == -1) {
			perror("Write failed\n");
			if (close(fd) == -1) {
				perror("Close failed\n");
				return errno;
			}
			return errno;
		}
		else {
			printf("Wrote %d bytes\n", num);
			if (fstat(fd, &st) == -1) {
				perror("Stat failed\n");
				return errno;
```



```c
			}
			/* this will not exactly print the FIFO size as per given condition
			 * since write by default is blocking and hence will wait for the
			 * read to finish and only then will statements ahead of it in this
			 * code execute
			 */
			//TODO find out how to get FIFO size immediately after write and
			//before the server reads it
			printf("Size of FIFO after client writing but before server reading: %d\n", (int)st.st_size);
		}
	}
	if (close(fd) == -1) {
		perror("Close failed\n");
		return errno;
	}
	return 0;
}
```

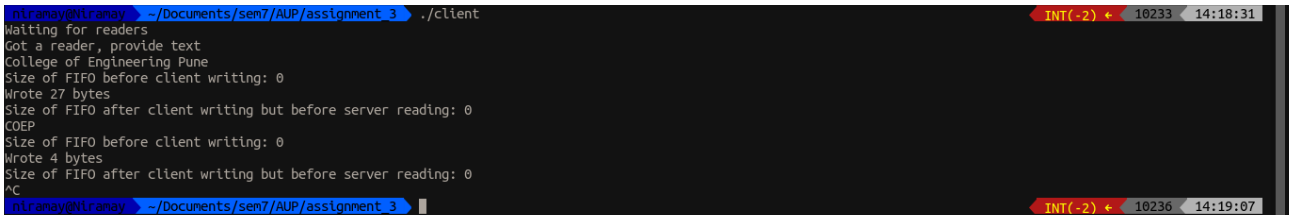
Server-



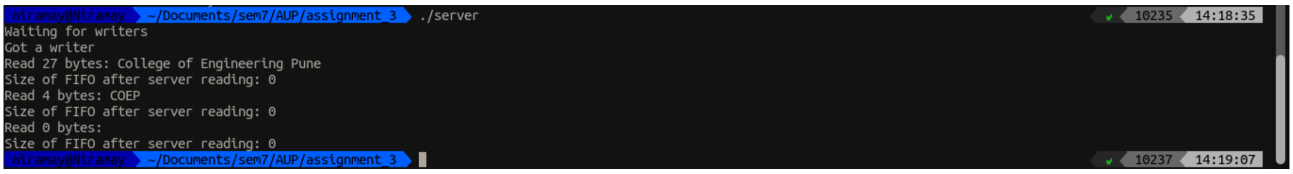
```c
#define FIFO_NAME "america"

int main(int argc, char *argv[]) {
    char s[100];
    int num, fd;
    struct stat st;
    printf("Waiting for writers\n");
    fd = open(FIFO_NAME, O_RDONLY);
    if (fd == -1) {
        perror("Open failed\n");
        return errno;
    }
    printf("Got a writer\n");
    do {
        if ((num = read(fd, s, 300)) == -1) {
            perror("Read failed\n");
            if (close(fd) == -1) {
                perror("Close failed\n");
                return errno;
            }
            return errno;
        }
        else {
            s[num] = '\0';
            printf("Read %d bytes: %s\n", num, s);
            if (fstat(fd, &st) == -1) {
                perror("Stat failed\n");
                return errno;
            }
            printf("Size of FIFO after server reading: %d\n", (int)st.st_size);
        }
    } while (num > 0);
    if (close(fd) == -1) {
        perror("Close failed\n");
        return errno;
    }
    return 0;
}
```

Output-



```
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  ./client                INT(-2)  10233  14:18:31
Waiting for readers
Got a reader, provide text
College of Engineering Pune
Size of FIFO before client writing: 0
Wrote 27 bytes
Size of FIFO after client writing but before server reading: 0
COEP
Size of FIFO before client writing: 0
Wrote 4 bytes
Size of FIFO after client writing but before server reading: 0
^C
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3                          INT(-2)  10236  14:19:07
```



```
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3  ./server                ✔  10235  14:18:35
Waiting for writers
Got a writer
Read 27 bytes: College of Engineering Pune
Size of FIFO after server reading: 0
Read 4 bytes: COEP
Size of FIFO after server reading: 0
Read 0 bytes:
Size of FIFO after server reading: 0
niramay@Niramay  ~/Documents/sem7/AUP/assignment_3                          ✔  10237  14:19:07
```

TODO- The current print in client will not exactly print the FIFO size as per given condition since by default write to FIFO is blocking and hence will wait for the read to finish and only then will statements ahead of it will execute. Hence, find out how to get FIFO size immediately after write and before the server reads it.