1. What are the differences between blocking/ non-blocking system call ? What are the difference between synchronized / asynchronized IO ? (2pts)

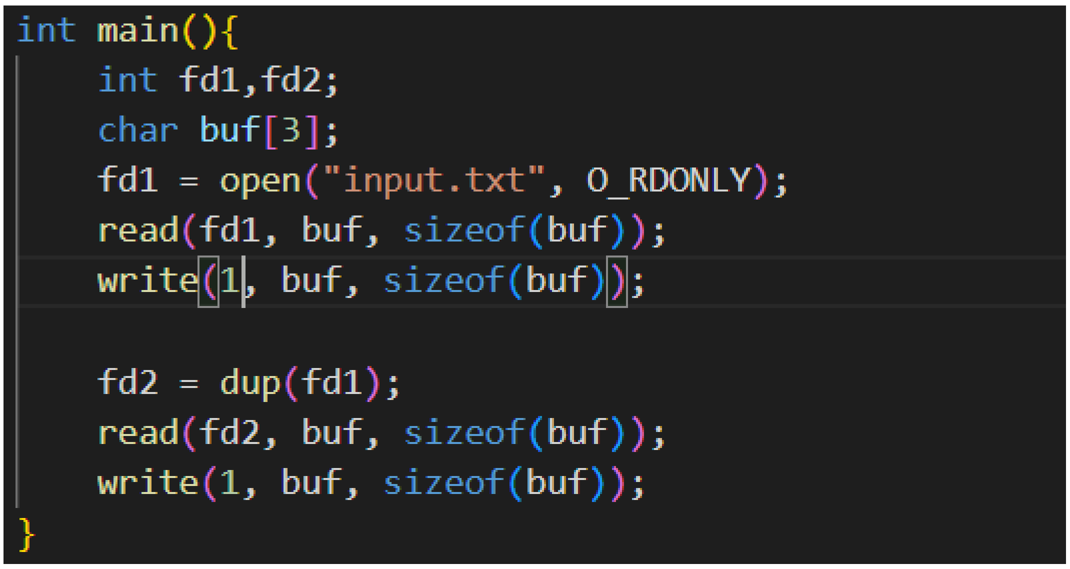
Ans:

- Blocking: About function call behavior

Blocking means that the program pause and wait the system call complete.
Non-blocking means that the program call the system call and do another task.If the system call complete, program will receive a signal

- Synchronization: About the data movement

Synchronized means that the data in the kernel buffer will synchronize to the hard disk. If the data in kernel buffer and hard disk are the same, it will return.
Asynchronized means that the data will write in the kernel buffer and write to hard disk later.

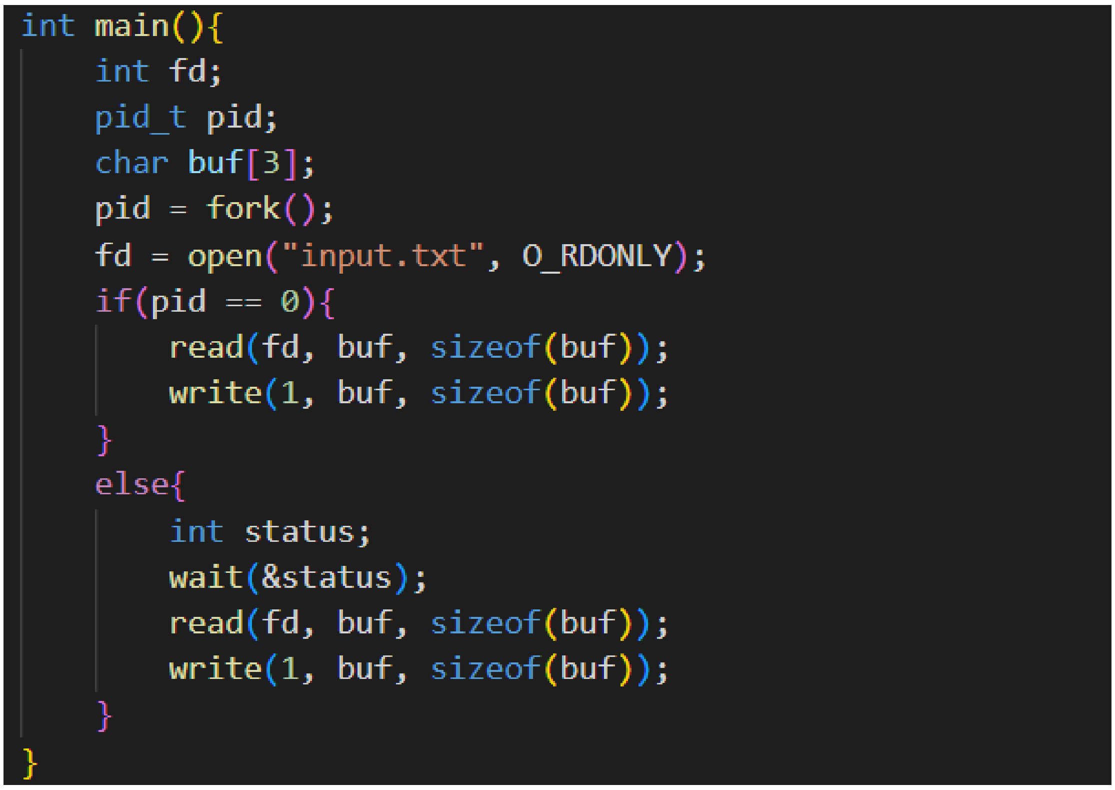


```c
int main(){
    int fd1,fd2;
    char buf[3];
    fd1 = open("input.txt", O_RDONLY);
    read(fd1, buf, sizeof(buf));
    write(1, buf, sizeof(buf));

    fd2 = dup(fd1);
    read(fd2, buf, sizeof(buf));
    write(1, buf, sizeof(buf));
}
```

2. Consider the code segment above, what will the content be in stdout after executing? (1pt) (“abcdef” in input.txt)

Ans: abcdef
Because two file pointers in the same program share the same file offset pointer. Hence, fd2 point to “d” at initial.



```
int main(){
    int fd;
    pid_t pid;
    char buf[3];
    pid = fork();
    fd = open("input.txt", O_RDONLY);
    if(pid == 0){
        read(fd, buf, sizeof(buf));
        write(1, buf, sizeof(buf));
    }
    else{
        int status;
        wait(&status);
        read(fd, buf, sizeof(buf));
        write(1, buf, sizeof(buf));
    }
}
```

3. Assume input.txt has only one line of characters “abcdef”, Consider the code segment above, what will the content be in stdout after executing? (2pts)

Ans: abcabc

Because the program open after fork(), so they use the different offset pointer to the same file. Hence, they read the same segment of the file.