**Operating System and Design (19CS2106A)**

**Advanced Lab- 5**

**XV6 file system checker:**

For this project, we will use the xv6 file system image as the basic image that we will be reading and checking. The file **include/fs.h** includes the basic structures you need to understand, including the superblock, on disk inode format (struct dinode), and directory entry format (struct dirent). The tool **tools/mkfs.c** will also be useful to look at, in order to see how an empty file-system image is created.

Make sure to look at **fs.img** , which is a file system image created when you make xv6 by the tool **mkfs** (found in the tools/ directory of xv6). The output of this tool is the file **fs.img** and it is a consistent file-system image

Steps:

1. $ git clone <https://github.com/gauthamsunjay/filesystemchecker.git>
2. $ cd filesystemchecker
3. $ ls

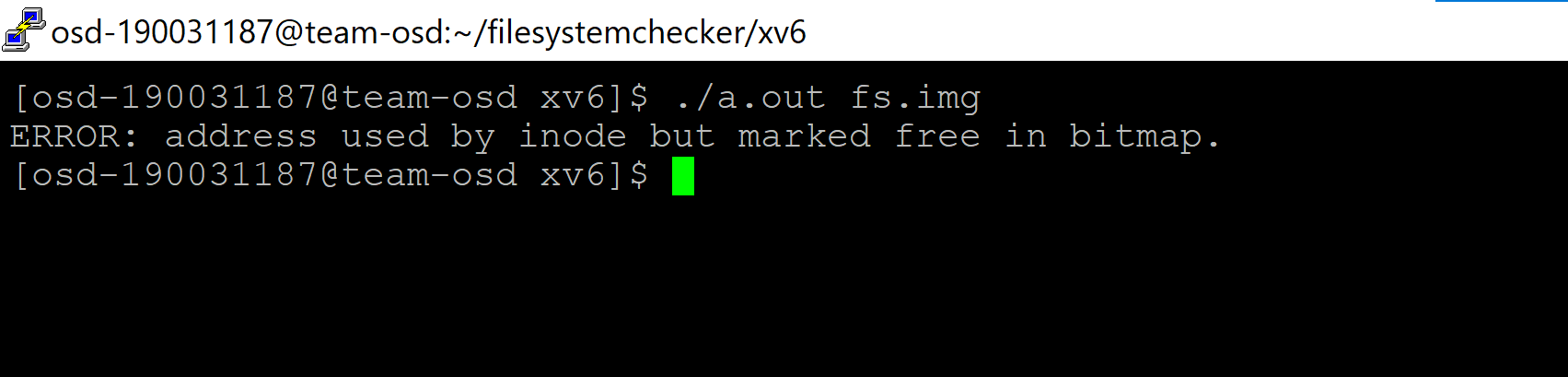
fs.h README.md xcheck.c

1. $ cc xcheck.c
2. $ git clone git://github.com/mit-pdos/xv6-public.git xv6
3. $ cd xv6
4. $ make qemu-nox
5. type ctrl a, x
6. $ cd ..
7. $ cp fs.h ./xv6
8. $ cp xcheck.c ./xv6
9. $ cd xv6
10. $ cc xcheck.c
11. $ ls

When we type ls we can see fs.img file which has been created

1. $ ./a.out fs.img

It prints “ERROR:address used by inode but marked free in bitmap” as one inode is not enabled.



**UNIX system programming**

1. dup2

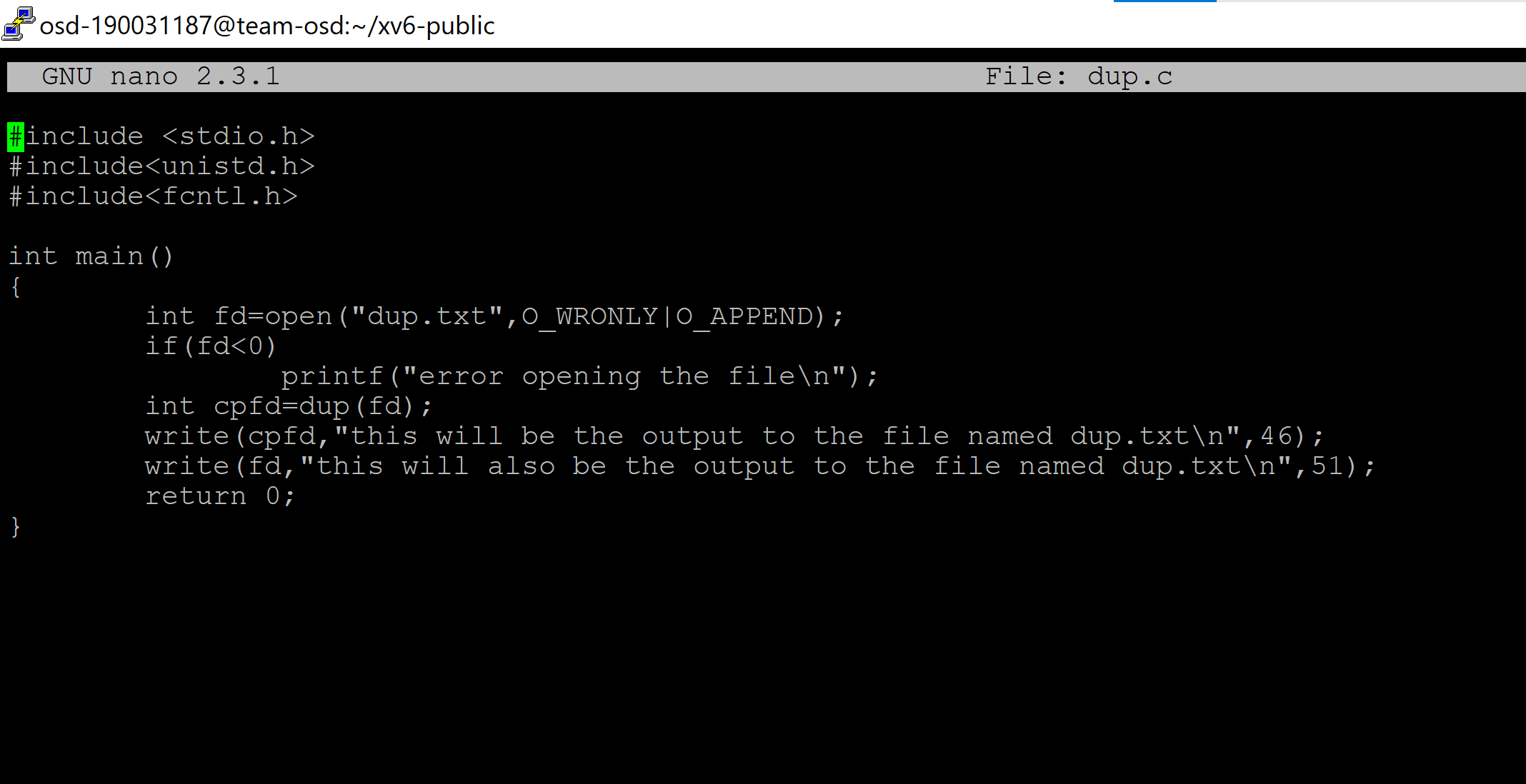
The dup() system call creates a copy of a file descriptor.

• It uses the lowest-numbered unused descriptor for the new descriptor.

• If the copy is successfully created, then the original and copy file descriptors may be used interchangeably.

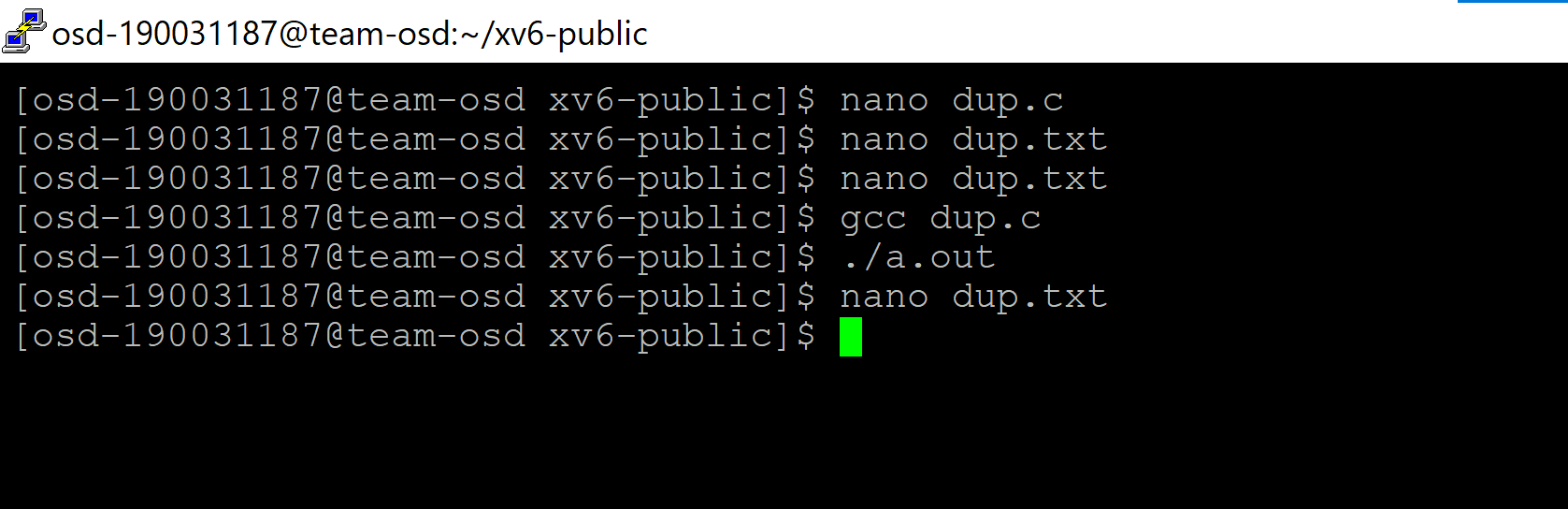
• They both refer to the same open file description and thus share file offset and file status flags.

dup.c

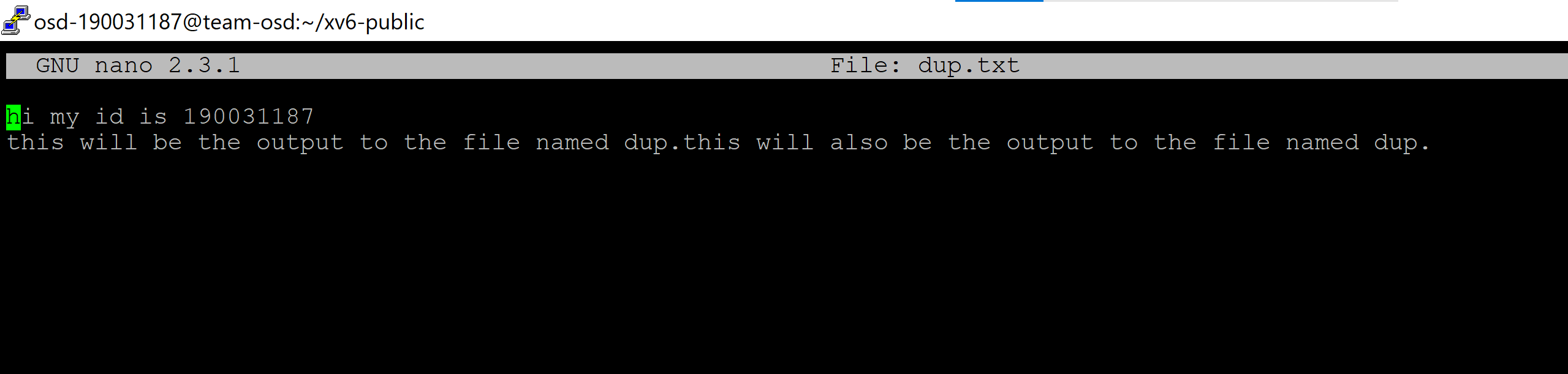


dup.txt:before running prog



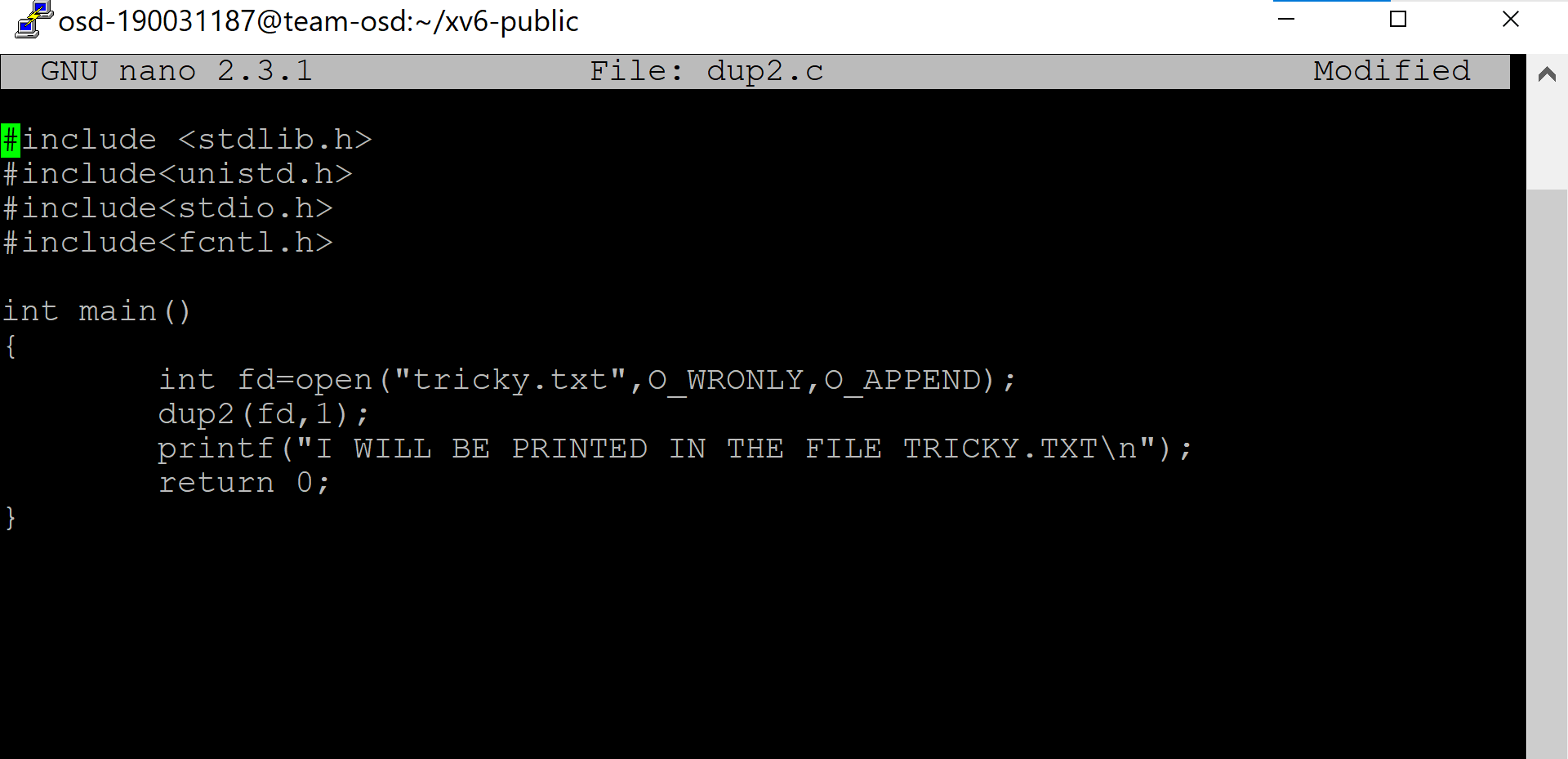


After running program : dup .txt consists



int dup2(int oldfd, int newfd); oldfd: old file descriptor newfd new file descriptor which is used by dup2() to create a copy.

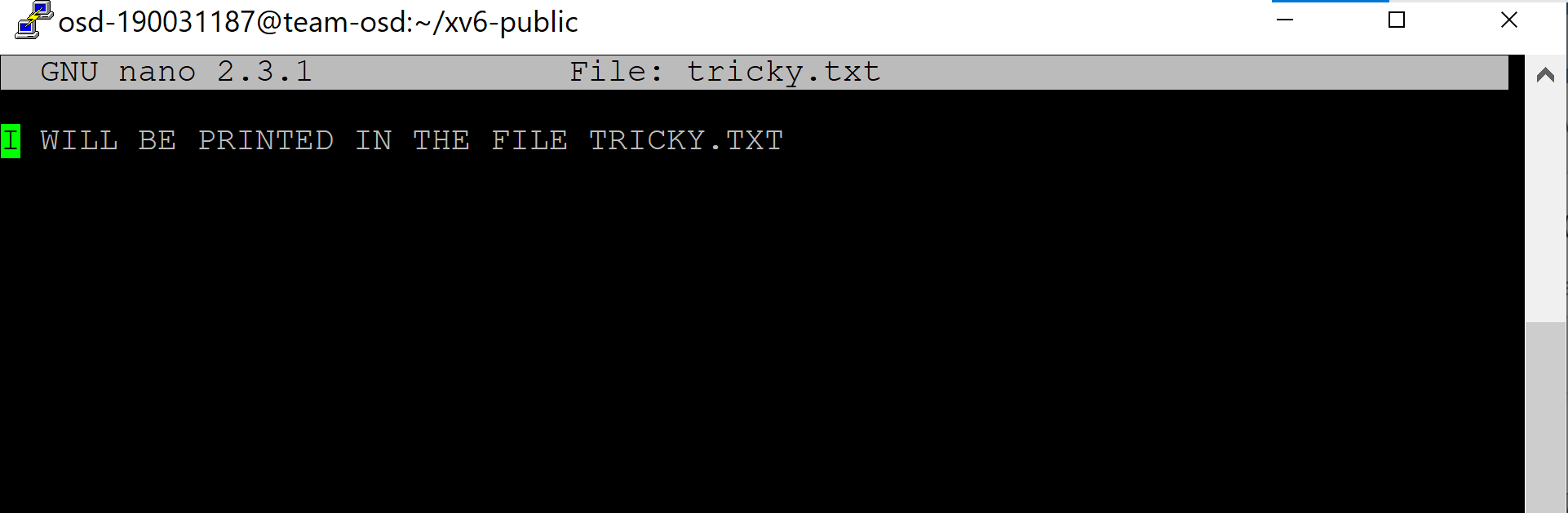
dup2.c



tricky.txt:before running prog



After running program: tricky .txt consists

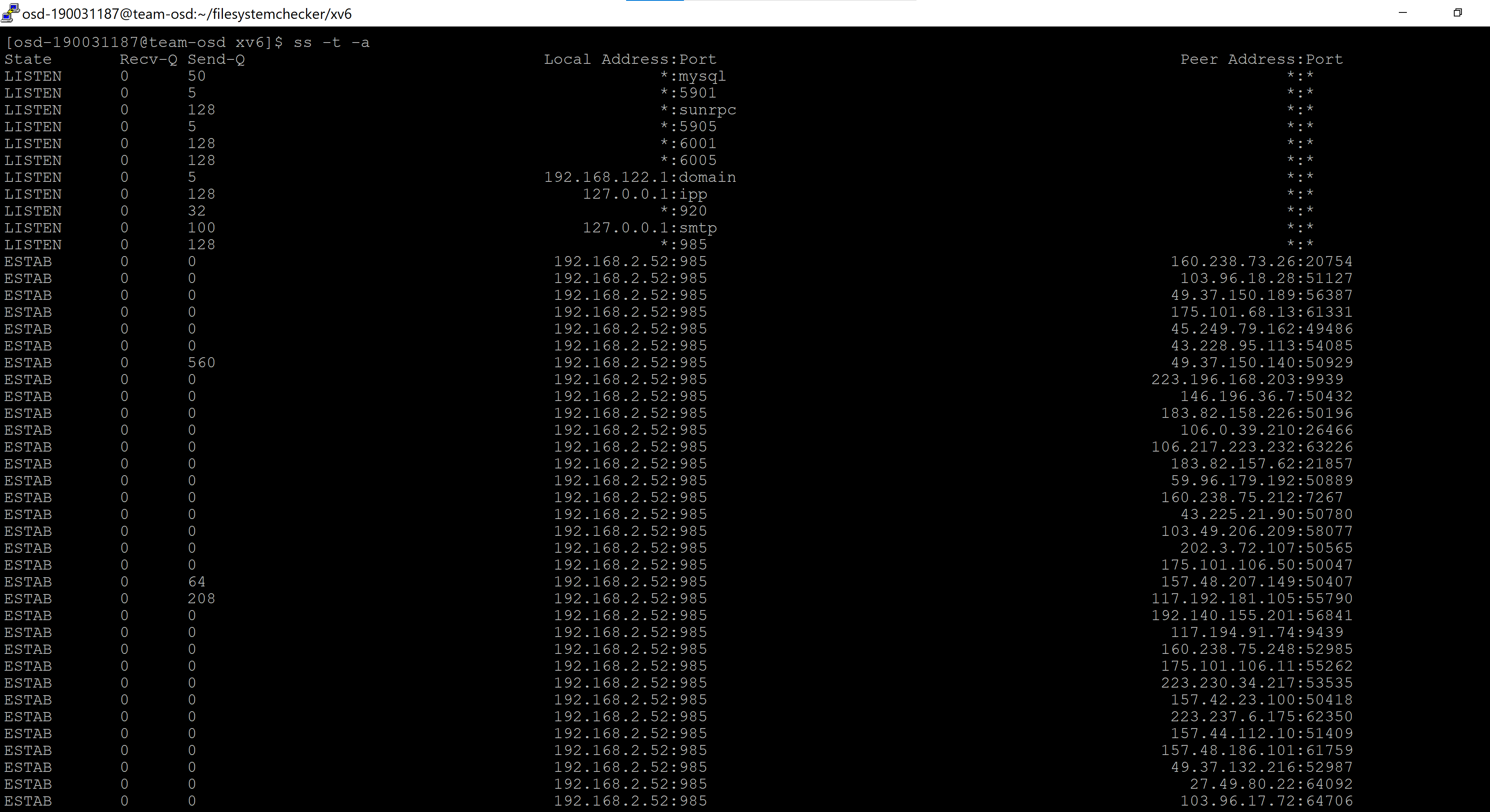


2. Unix sockets: TCP, UDP,

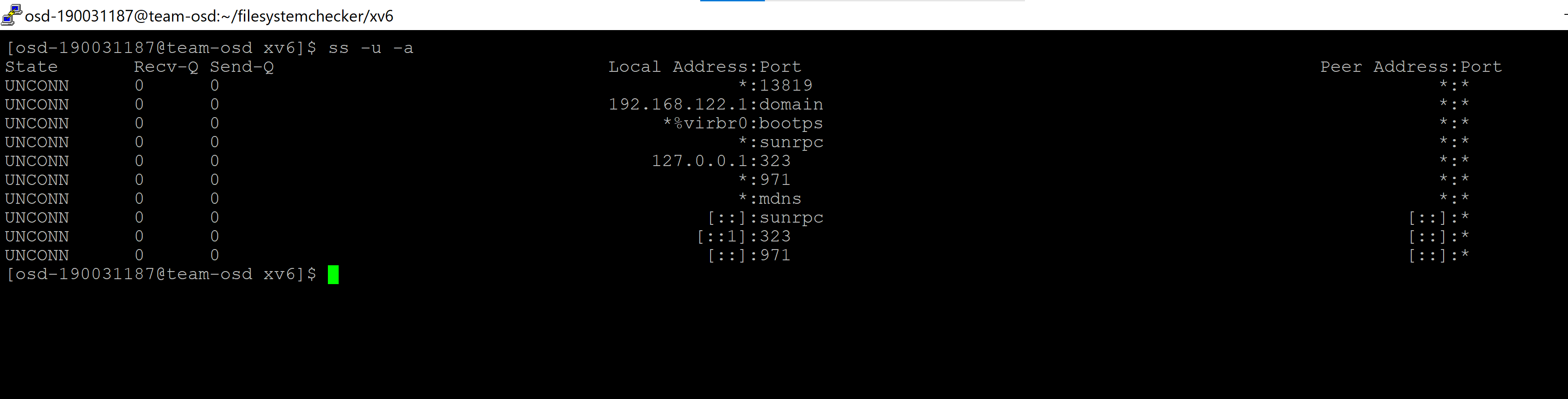
It is set to the protocol used. Internet services are provided using either TCP or UDP.

TCP:

DISPLAYS ALL TCP SOCKETS USING: $ss -t -a command



DISPLAYS ALL UDP SOCKETS USING: $ss -u -a command



3. Demonstrate How RPC Works: invoking square remote procedure with arguments. Invoking date remote procedure without argument

A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

Some of the advantages of RPC are as follows −

* Remote procedure calls support process oriented and thread oriented models.
* The internal message passing mechanism of RPC is hidden from the user.
* The effort to re-write and re-develop the code is minimum in remote procedure calls.
* Remote procedure calls can be used in distributed environment as well as the local environment.
* Many of the protocol layers are omitted by RPC to improve performance. Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows −

* The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
* There is no flexibility in RPC for hardware architecture. It is only interaction based.
* There is an increase in costs because of remote procedure call.