

# Unix Domain Sockets project

## ***Project description***

Although we have sockets for PF\_INET, we don't have support for PF\_UNIX sockets otherwise known as Unix Domain Sockets or POSIX Local IPC Sockets. This project involves writing the code necessary to implement them. In order to successfully complete this project, it is required to have intimate knowledge of the socket API and a good understanding of the MINIX 3 architecture.

## ***Sheep-Goat project***

Before attempting to apply for this project, you are required to successfully finish the sheep-goat project. How well you answer the following questions helps us determine how good a sheep you are (the best sheep will be picked to do this project).

1. In a sense, UDS resemble pipes. Explain in detail how in MINIX 3 pipes are created. For example, a user process invokes the pipe(2) system call which prepares a message and sends it to VFS where...etc. Include details about the internals of VFS.
2. Describe in detail how reads and writes from/to pipes work. How much data can a process read or write? When does it block? Which servers are involved? How do they cooperate? Describe the message flow between all parties involved. Diagrams are useful!
3. What happens currently when a process does a write(2) on a socket (PF\_INET socket, as we don't have other sockets at the moment). What happens when a process calls send(2) on a socket?
4. This is a programming assignment which allows us to assess your programming skills and socket API knowledge. In MINIX 3 we have a POSIX test set under /usr/src/test. You are asked to implement a test program that will verify the UDS implementation you are going to write. Of course, you can't actually compile and run the program on MINIX, so write it on Linux. If everything went fine, it prints "ok\n", else it prints an error message on each error and a total amount of error encountered. See other tests in that directory for examples.

The program should test every aspect of UDS: the existence of API functions, macros, constants and the working of UDS itself. It should test whether it works under normal conditions where the API is used exactly as intended, but also the corner cases where incompatible or wrong values are used. For example, a server invokes socket(2), bind(2), listen(2), accept(2), send(2), recv(2), and close(2). What happens if bind(2) is not invoked before calling listen(2)? Are the correct error messages returned? POSIX 2008 describes exactly what must happen:

<http://www.opengroup.org/onlinepubs/9699919799/>

Do not underestimate the difficulty of this assignment.

Completing this project requires you to dive into the MINIX internals involved with the project and read the POSIX specification in order to write a good test program. The test program allows you to properly test your UDS implementation.