B4 - Network Programming

B-NWP-400

Bootstrap

myTeams

0.1

Bootstrap

language: C

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

This bootstrap is an introduction for the myTeams project.
Please note that Jenkins tests are not available for now. download the bootstrap folder from the Intranet.
Note that for the following steps you should not use global or static variables
Handle all the error gracefully (at least a perror, preferably without exit).

UNDERSTAND SELECT

The `select` function is used to monitor File Descriptors, please read **man 2 select** and take a look at the wiki.

+ +STEP1: READ AND SELECT

In this exercise, you should always have the following output for every tests.
Please complete the `main.c` file with your own code.
You can launch a set of tests with `./tests.sh` the program will stop if a test fail and display some debug information (if stdin is closed stop the program).
Take a look inside the script you understand what is tested.
You can also take a look inside `on_command.c` but do not edit it.

```
> (echo -ne "hellon"; sleep 0.5; echo -ne "worldn") | ./bin
COMMAND: "hellon00000000000000000000000000"
OK: the command seem all right.
COMMAND: "worldn00000000000000000000000000"
OK: the command seem all right.
```

Here some examples of data that could be send to on_command please; look at errors that are shown.

```
{
  char cmd[MAX_COMMAND_LENGTH + 1] = "hellon";
  on_command(cmd);
}
{
  char cmd[MAX_COMMAND_LENGTH + 1] = "hello";
  on_command(cmd);
}
{
  char cmd[MAX_COMMAND_LENGTH + 1] = "hellonn";
  on_command(cmd);
}
{
  char cmd[MAX_COMMAND_LENGTH + 1] = "hellonhel";
  on_command(cmd);
}
{
  char cmd[MAX_COMMAND_LENGTH + 1] = "hellon00hel";
  on_command(cmd);
}
```

You need to reset every fd_set before calling select.

You may need a temporary buffer to store the data until you receive a complete command

+ +Step2: Write and select

Compile this main and start it with strace.

```
# include <sys/select.h>
# include <stddef.h>

int main(void) {
  fd_set read_fds;
  fd_set write_fds;
  fd_set except_fds;
  do {
    FD_ZERO(&read_fds);
    FD_SET(1, &write_fds);
    FD_ZERO(&except_fds);
  } while (select(4, &read_fds, &write_fds, &except_fds, NULL) != -1);
}
```

If you need more information about the fds look at **man 2 strace**, and find the flag --`decode-fds=set`.

As you can see something interesting append !
The select never stop unlocking... which is not really nice for our cpu and battery consumption (#Green-Code)
This is because the fd 2 (stdout) is most of the time ready to be write to.
From now on try to create a program that read when needed from `stdin` and when a line break is detected (or the command buffer is full) write the content of the command (only the command not all the buffer) in UPPER case on `stdout`.
Please remember to check if the fd is ready to be read or write before calling read or write.

create a datatype that contain:
- an input fd.
- an output fd.
- one buffer for input.
- one buffer for output.

+ +Step3: Socket and select

Finally try to create a server that do the same thing as the step 2 but with sockets.
This server must be able to handle multiple clients at the same time (without fork).
This time, if the in_buffer is full close the connection with the client and display an error on `stderror`.

Allocate all the resource needed by the client at it connection.

Remember about `nc` (man nc)