

# CSIS0234B Computer and Communication Networks (Class B)

## Tutorial 1

### POP3 client

In this tutorial, we write a program to talk to a POP3 server using the socket interface.

#### 1. POP3 Protocol

The POP3 protocol allows you to check whether there are E-mails in your mailbox, and retrieve the E-mails there and delete them. It runs on the well-known TCP port number 110. The server starts the dialog by sending a message saying that it is talking POP3. From then on, the client sends commands to the server, the server executes it and provides responses. Messages are terminated by the CR-LF (ASCII 13 and then 10) sequence. We only use the following commands:

*USER login*

*USER* is the first command sent by the client, to specify the login id. of the mailbox to be manipulated. Expect a boolean response (see below).

*PASS password*

The client then use the *PASS* command to provide a password. Once this is completed successfully the following commands may be used. Expect a boolean response.

*STAT*

Ask for the status of the mailbox. The response looks like `+OK 2 1000`, which tells that 2 messages of 1000 bytes in total are in the mailbox.

*QUIT*

Terminate the connection. Expect a boolean response.

A boolean response is either `+OK` (positive) or `-ERR` (negative), both may be followed by some descriptive messages to tell what is happening. A POP3 dialog looks like this:

```
+OK POP3 study-172 v2003.83 server ready
USER tmchan
+OK User name accepted, password please
PASS abcd1234
+OK Mailbox open, 6 messages
STAT
+OK 6 9848
QUIT
+OK Sayonara
```

We will write a program to perform the above interactions. You can test it by hand, using `telnet study.csis.hku.hk pop3`. Here `study.csis.hku.hk` is our POP3 server. If you do that, first login `virtue` and change your password temporarily: `telnet` shows your password as you type since it doesn't know that it is a password.

#### 2. Your task

Write a C or C++ program that performs the above interactions. It should ask for a login and password for use in the connection. If any command or system call fails, print an error message and exit. After the *STAT* command returns, print the number of messages in the mailbox and

the total size that is responded by the server. The program thus runs like this:

```
> ./pop3client
Login: tmchan
Password: (hidden)
Maildrop: 6 9848
```

Here are the steps involved:

1. Write a program to connect to the POP3 server. The code should be similar to that you can find in the tutorial reading. If any system call fails, output an error message describing the error (using `strerror()` and `gai_strerror()`).
2. Check that the connection is made successfully: Add a `sleep(1000);` call after making your connection, run your program, and list the directory `/proc/<pid>/fd` (which is a listing of file descriptors for process `<pid>`) to check that a socket is created. Use `netstat -te` to read an extended (e) listing of all TCP connections (t) of the system to make sure that the connection is already made. Also try adding the `-n` flag to see the actual, numerical addresses. Note the local port number used by the connection.
3. Add a function to read a line from the connection, terminated by CR-LF. This can be done using a function similar to the following.

```
void read_response(int fd, char buf[1024]) {
    char ch;
    int len = 0;
    for (;;) {
        if (read(fd, &ch, 1) == -1) /* read chars one by one */
            die("Read");
        if (ch == '\n')
            break;
        if (len < 1023) /* don't overflow the buffer */
            buf[len++] = ch;
    }
    if (buf[len-1] == '\r')
        --len;
    buf[len] = 0; /* make it a C-style string */
}
```

In your main function, call the above function to get a line and print it out. Check whether it is what you expect (i.e., the server ready message).

4. Add a function to send a command to the server and wait for its response. Check whether the response is positive or negative, by checking its first character. If an error occurs, the function should terminate the program after printing the error message, which can be found by skipping the first 5 characters of the response (i.e., "-ERR "), e.g., by `printf("%s\n", response+5)`. Modify the main function to make the program asks for the login and password. Note that you can read a line without echoing using `getpass("Password: ");`, where `getpass` is a function defined in `<unistd.h>`. See its man page.