Documentation of the application layer of the first project for IPK 2014/2015
Name and surname: Jakub Vitásek
Login: xvitas02

1 Assignment

The assignment was to create a process–driven client–server program using TCP and sockets API. The application layer carries out the main functionality of searching for and transmitting client–specified entries from the UNIX file /etc/passwd.

2 Implementation

The client parses the user–side arguments using `getopt()` and stores the options in variables. If an option was specified, the respective variable is set to 1. Otherwise, it maintains the value of 0. Also, variables `is_login` and `is_uid` serve as an indicator for the server to know what to request. All the integer variables of options plus the indicator variables are sent to the server in this form:

```
snprintf(argsbuffer, 9, "%d%d%d%d%d%d%d%d", L, U, G, N, H, S, is_login, is_uid);
```

Once the server receives this stream of 9 integers, it knows what to look for in the passwd file. Now, depending on the content of the indicator variables, the server receives the login/uid value to the appropriate placeholder and acknowledges the client. At this point, the server has all the information needed to start searching for the requested entries.

2.1 Getting the requested /etc/passwd entries

Utilizing the function `getpwnam()` and `getpwuid()` from `<pwd.h>`, the complete information is stored in a variable `struct passwd *pwd`, a structure containing different parts of the passwd file delimited by the colon sign. To make up for these functions being case–sensitive (the assignment demands case–insensitivity), the login is converted to lowercase. Based on what the client requested (L/U/G/N/H/S options), a result string is composed. Either both UID/GID is included, only GID or UID, or neither.

Furthermore, the string contains substrings like name, home directory or shell. These are either empty strings (the client did not request those options), or they contain the needed information. This is how the string is composed when both UID and GID were requested:

```
snprintf(res_buff, sizeof(res_buff), "%s %u %u %s %s %s", name, uid_fin, gid_fin, gec, home, sh);
```

2.2 Receiving the results and validating

Based on how the received data is delimited (‘&’ marking the beginning of the result string, ‘;’ marking the end of the string), the client program is able to print the requested passwd info in the correct format. If nothing is found, the server returns a message informing the user the matching entry wasn’t found.

2.3 Exit values

There are 3 exit codes in this program. Their purpose is to help the user/developer better understand the origin and character of an error. All types of return are listed below.

- Exit code `“0”` – There was no error. Both client and server side of the program ran successfully.
- Exit code `“1”` – There was an error in the connection or socket handling, sending, receiving or closing a socket.
- Exit code `“2”` – There was an error in the user–side arguments, that includes unknown option or too little arguments.