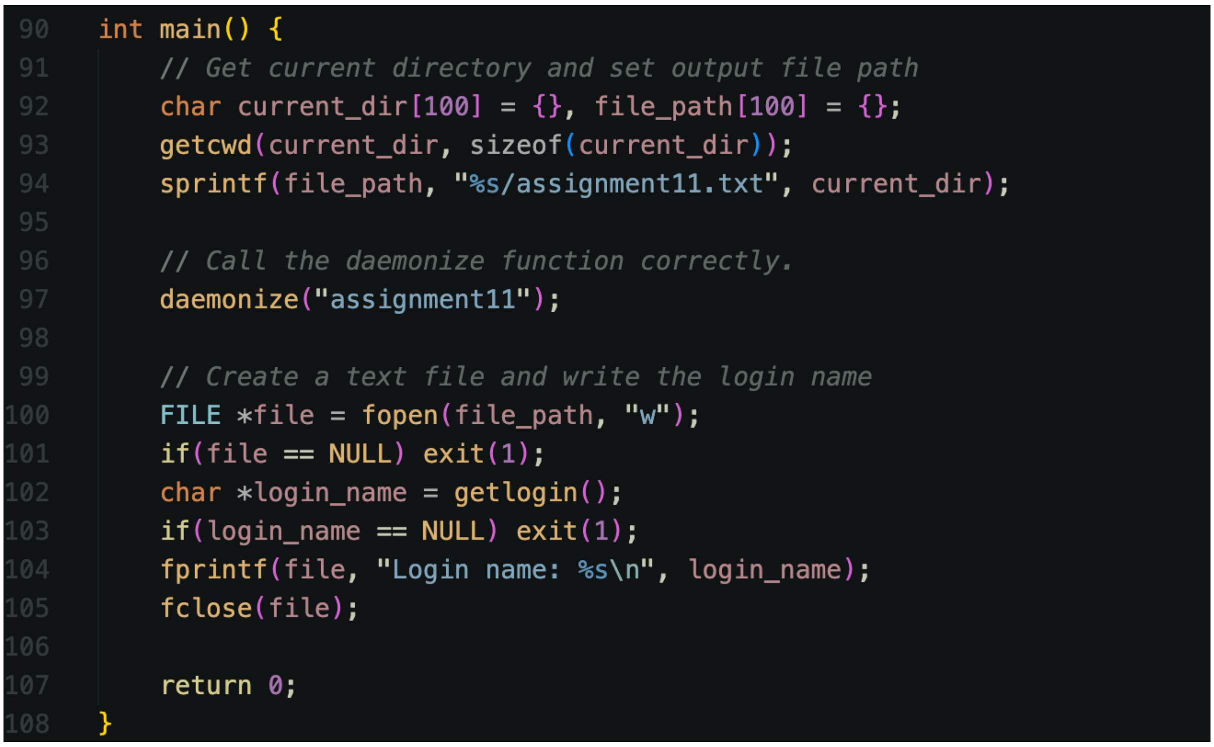
Unix Assignment11

Team19 108072244 邱煒甯 108032053 陳凱揚 112065525 簡佩如

Implementation

Before calling the `daemonize` function, we get the current working directory with `getcwd` function. Then, we set the output file path because the `daemonize` function will switch the directory to `/`.

After calling the `daemonize` function, we open the output file in write mode and get the `login_name` with `getlogin` function. We then write this information to the output file.



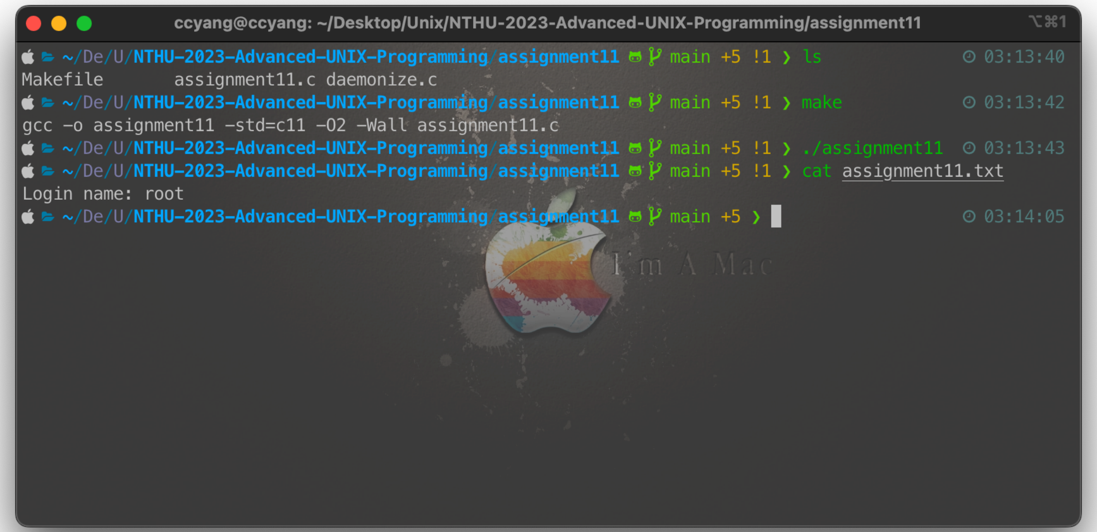
```c
int main() {
    // Get current directory and set output file path
    char current_dir[100] = {}, file_path[100] = {};
    getcwd(current_dir, sizeof(current_dir));
    sprintf(file_path, "%s/assignment11.txt", current_dir);

    // Call the daemonize function correctly.
    daemonize("assignment11");

    // Create a text file and write the login name
    FILE *file = fopen(file_path, "w");
    if(file == NULL) exit(1);
    char *login_name = getlogin();
    if(login_name == NULL) exit(1);
    fprintf(file, "Login name: %s\n", login_name);
    fclose(file);

    return 0;
}
```

Result



```
ccyang@ccyang: ~/Desktop/Unix/NTHU-2023-Advanced-UNIX-Programming/assignment11
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment11  main +5 !1 › ls            03:13:40
Makefile      assignment11.c daemonize.c
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment11  main +5 !1 › make          03:13:42
gcc -o assignment11 -std=c11 -O2 -Wall assignment11.c
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment11  main +5 !1 › ./assignment11  03:13:43
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment11  main +5 !1 › cat assignment11.txt
Login name: root
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment11  main +5 ›                  03:14:05
```

Question

- Explain the purpose of every step executed in the `daemonize` function.

  1. `umask` : Clear file creation mask.

     Daemons have to create files with exactly the permissions they specify, so they reset the mask.

  2. `getrlimit` : Get maximum number of file descriptors.

     This fetches the system limit for file descriptors for this process. It's used later to close all file descriptors.

  3. `fork` , `setsid` : Become a session leader to lose controlling TTY.

     The initial fork() allows the parent process to exit and ensures that the child process (the future daemon) is not a process group leader. The setsid() call creates a new session and makes the child process the session leader, which detaches it from the terminal (losing the controlling TTY).

  4. `sigaction` : Ensure future opens won't allocate controlling TTYs.

     The process sets the disposition of SIGHUP to ignore. This is because when a session leader that has a controlling terminal terminates, the system sends SIGHUP to this session leader.

  5. `chdir` : Change the current working directory to the root so we won't prevent file systems from being unmounted.

    This is done to ensure that the daemon does not prevent file systems from being unmounted.

6. `close` : Close all open file descriptors.

   The daemon closes all open file descriptors which are not unnecessary for the daemon's operation.

7. `dup` : Attach file descriptors 0, 1, and 2 to /dev/null.

   Since the daemon has closed all of its file descriptors, it attaches `stdin`, `stdout`, and `stderr` to `/dev/null` to ensure that any library calls that assume these are open will not fail.

8. `openlog`, `syslog` : Initialize the log file.

   This initializes the system logger, allowing the daemon to write to the system logs.

- Discuss what would happen to the process after becoming a daemon process.
  - The process has no controlling terminal, which means it cannot be directly interrupted with keyboard inputs like CTRL+C.
  - It runs in the background and is detached from the user's session, meaning the user can log out, and the daemon will continue to run. It usually performs tasks without direct user interaction.
  - The daemon should have a mechanism to be stopped, usually through a signal like `SIGTERM` or by a control command to the process.