Homework #4

School ID: 201624476 Name: 박상운

Implement the logwatchdog daemon (never die…) program that will read / write the specific directory and its subdirectories.

(Writing a daemon in C (or daemonize a process) for absolute newbie:
https://blog.abhi.host/blog/2010/03/09/writingdaemon-
in-c-or-daemonize/)

The SystemInfoIO program will do the following sequentially.

1. Make a temporary directory file starting with your account under "/tmp".
2. Under the temporary directory file you just created, create a temporary file starting with "cse" .
3. Whenever a new user logs in the system, detect the login (use utmp(x) / wtmp) and write login user name and login time
to your temporary file (Use fread() / fwrite()).
4. Whenever the user logs out the system, detect the logout and then append logout time right after the history information
you wrote at step 3 to your temporary file (Use fread() / fwrite()).
5. Every time 30 seconds, your logwatchdog daemon will display the current history log data saved in your temporary file,
which was created at step 2 (Please try it. This is very interesting programming skills because the daemon process does
not have stdin, stdout, and stderr). If you cannot display the current history log data saved in your temporary file on the
monitor, you can use syslog and tail -f commands.

Homework #4

School ID: 201624476 Name: 박상운

1. Submit your program source with detailed description (comments)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <utmp.h>
#include <time.h>
#include <string>
#include <map>
using namespace std;

int main(void) {

    // log file related constants and variable
    const char LOGDIR_PATH[] = "/tmp/201624476";
    const char LOG_PATH[] = "/tmp/201624476/cse201624476";
    FILE *fStream;

    // utmp structure pointer variable
    struct utmp *utx;

    // time-related variables
    time_t the_time;
    struct tm *tm_ptr;
    char now_local_time[50];

    // check if log file directory exists
    if(access(LOGDIR_PATH, R_OK | W_OK) != 0) {
        // make logfile directory
        if(mkdir(LOGDIR_PATH, 0777) == -1 && errno != EEXIST) {
            fprintf(stderr, "directory create error: %s\n",
strerror(errno));
            return -1;
        }
    }

    // make logfile
    fStream = fopen(LOG_PATH, "a+");
    fclose(fStream);

    /* Our process ID and Session ID */
    pid_t pid, sid;

    /* Fork off the parent process */
    pid = fork();
    printf("pid = [%d] \n", pid);
    if (pid < 0) {
        exit(EXIT_FAILURE);
    }
```

Homework #4

School ID: 201624476 Name: 박상운

```
/* If we got a good PID, then
    *         we can exit the parent process. */
    if (pid > 0) { // Child can continue to run even after the parent has
finished executing
        exit(EXIT_SUCCESS);
    }

    /* Change the file mode mask */
    umask(0);

    /* Open any logs here */

    /* Create a new SID for the child process */
    sid = setsid();
    if (sid < 0) {
        /* Log the failure */
        exit(EXIT_FAILURE);
    }

    /* Change the current working directory */
    if ((chdir("/")) < 0) {
        /* Log the failure */
        exit(EXIT_FAILURE);
    }

    /* Close out the standard file descriptors */
    // Because daemons generally dont interact directly with user so there
is no need of keeping these open

    close(STDIN_FILENO);

    /* Daemon-specific initialization goes here */
    printf("starting daemon...\n");

    // map checks login state of users
    map <string, int> user_state;

    /* An infinite loop */
    // output routime period
    int cnt = 15;
   while (1) {
        char log_buff[40];
        /* Do some task here ... */
        // initialize logged-in users
        for(auto& c : user_state) {
            c.second = -1;
        }

        // set file stream
        fStream = fopen(LOG_PATH, "a");

        // initialize utent
        setutent();

        // loop utmp
        while((utx = getutent()) != NULL) {
```

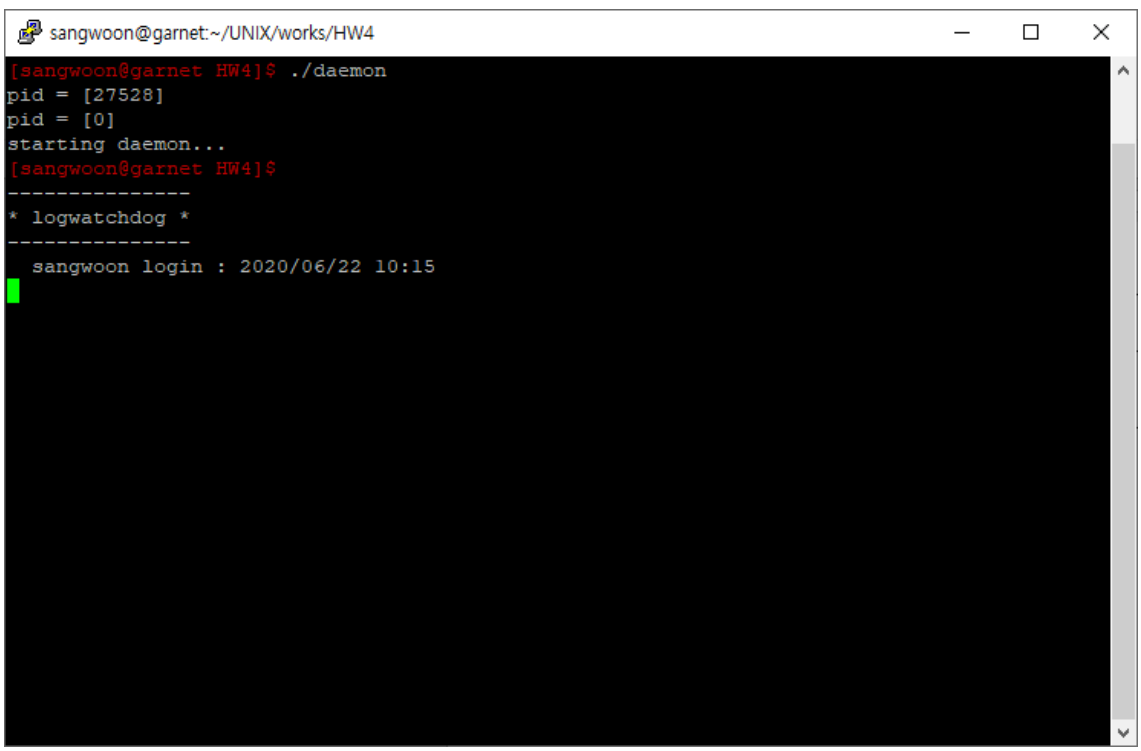
Homework #4

School ID: 201624476 Name: 박상운

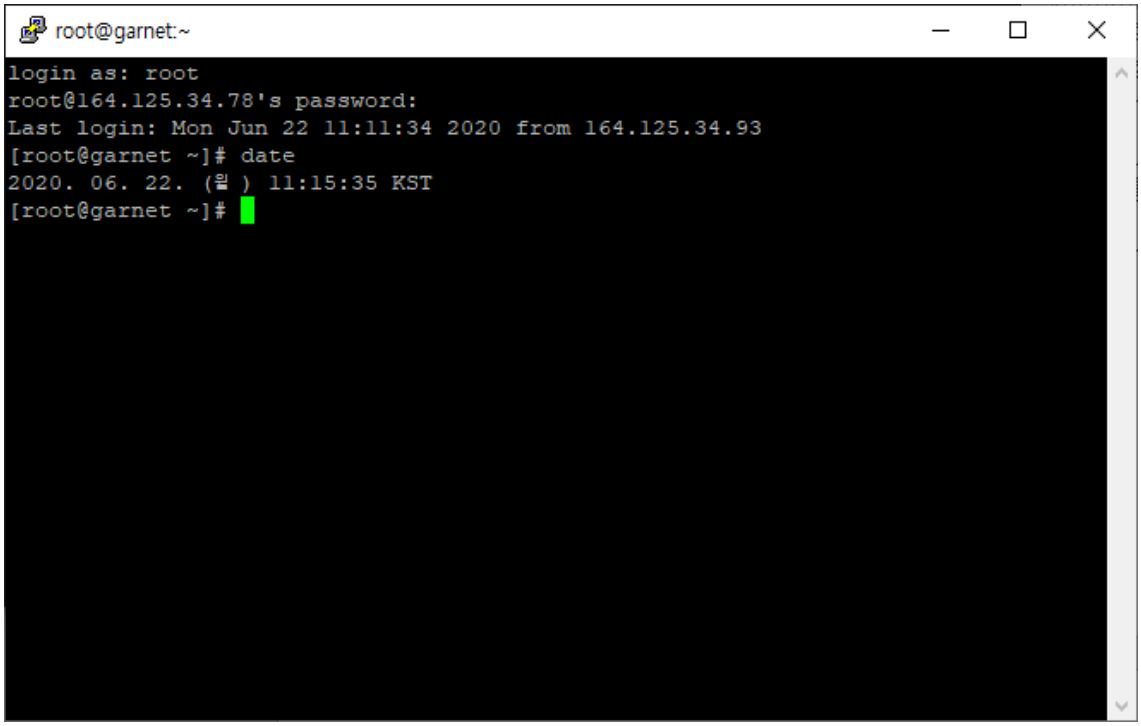
```
            // check if it is user process
            if(utx->ut_type != USER_PROCESS) continue;

            // get time information to now_local_time
            the_time = utx->ut_time;
            tm_ptr = localtime(&the_time);
            sprintf(now_local_time, "%d/%02d/%02d %02d:%02d", tm_ptr-
>tm_year+1900, tm_ptr->tm_mon+1, tm_ptr->tm_mday, tm_ptr->tm_hour, tm_ptr-
>tm_min);

            // if new user is logged in, write the login log
            if(user_state[string(utx->ut_name)] == 0) {
                sprintf(log_buff, "%10s login : %s\n", utx->ut_name,
now_local_time);
                fwrite(log_buff, strlen(log_buff), 1, fStream);
                user_state[string(utx->ut_name)] = 1;
            }
            // if user exists in user_state, check the flag value of it
            else {
                user_state[string(utx->ut_name)] = 1;
            }
        }

        // get current time information to now_local_time
        time(&the_time);
        tm_ptr = localtime(&the_time);
        sprintf(now_local_time, "%d/%02d/%02d %02d:%02d", tm_ptr-
>tm_year+1900, tm_ptr->tm_mon+1, tm_ptr->tm_mday, tm_ptr->tm_hour, tm_ptr-
>tm_min);

        // loop user_state
        for(auto c : user_state) {

            // if value of user is -1, it means username aren't contained
in utmp, so the user is logged-out
            if(c.second == -1) {
                char ch[100];
                strcpy(ch,(c.first).c_str());

                // write the logoff log
                sprintf(log_buff, "%10s logout : %s\n", ch,
now_local_time);
                fwrite(log_buff, strlen(log_buff), 1, fStream);
                user_state.erase(c.first);
            }
        }
        fclose(fStream);

        sleep(2); /* wait 2 seconds */

        // display logfile every 30 seconds (2 sec * 15 cnt)
        if(cnt == 15) {
            /* display logwatchdog every 30 secs */
            printf("\n---------------\n");
            printf("* logwatchdog *\n");
            printf("---------------\n");
```

```
            // check the size of log file
            fStream = fopen(LOG_PATH, "r");
            fseek(fStream, 0, SEEK_END);
            int lSize = ftell(fStream);
            rewind(fStream);

            // allocate the buffer and display the log file
            char* buffer = (char*)malloc(sizeof(char) * lSize);
            fread(buffer, 1, lSize, fStream);
            printf("%s", buffer);
            fclose(fStream);
            free(buffer);
            cnt = 0;
        }
        else  {
             cnt++;
        }
    }
    fclose(fStream);
    exit(EXIT_SUCCESS);
}
```

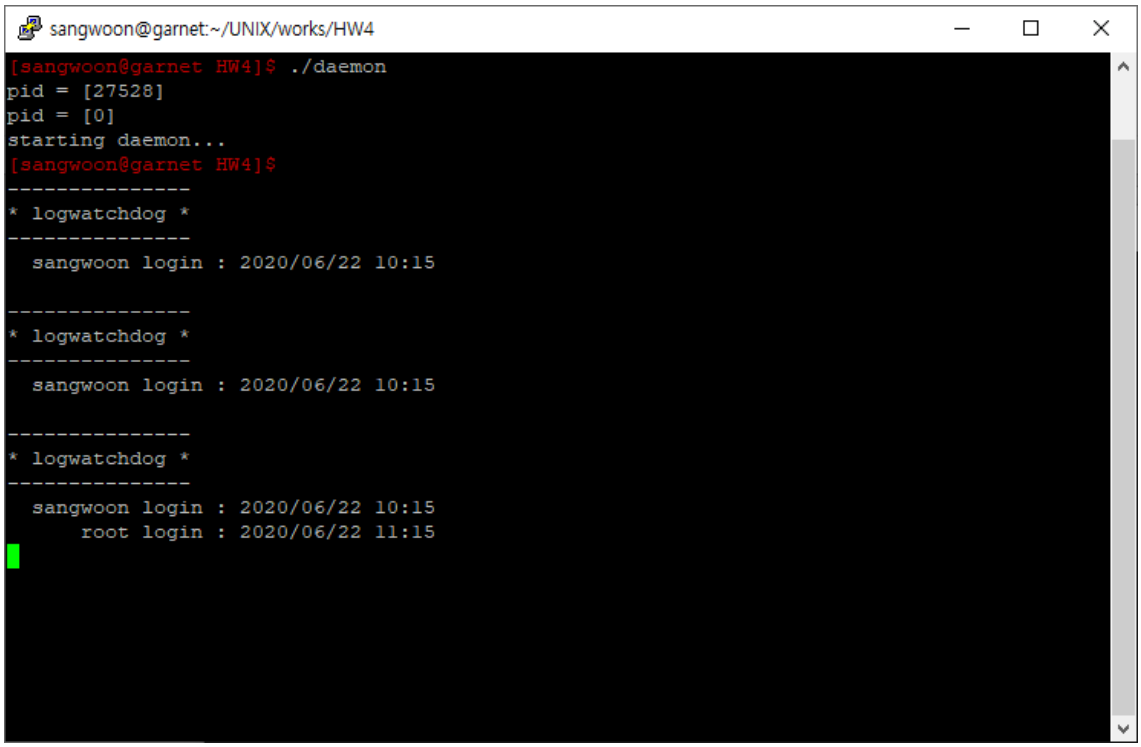
2. Put a screen shot of output generated by your program as here.



This is an initial start display of logwatchdog daemon. I've already logged-in with my account 'sangwoon'.



And then, I've logged-in with 'root' account.



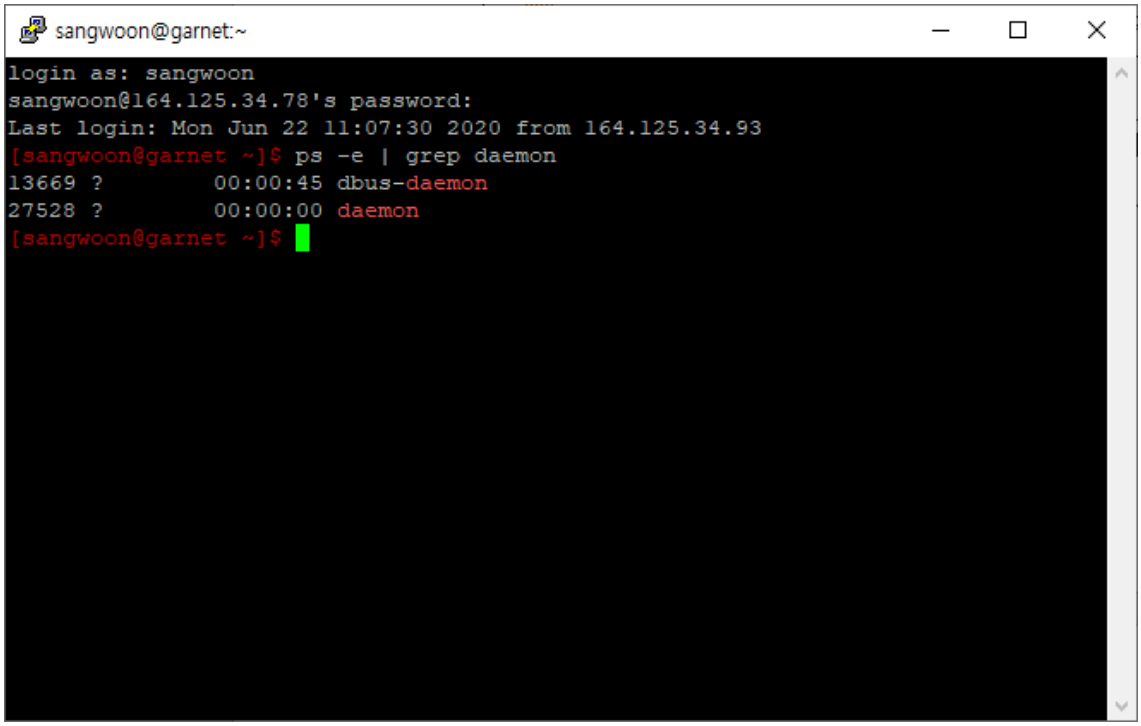
The logwatchdog daemon displays login information of 'root' account.

```
root@garnet:~
login as: root
root@164.125.34.78's password:
Last login: Mon Jun 22 11:11:34 2020 from 164.125.34.93
[root@garnet ~]# date
2020. 06. 22. (월 ) 11:15:35 KST
[root@garnet ~]# date
2020. 06. 22. (월 ) 11:16:47 KST
[root@garnet ~]# exit
```

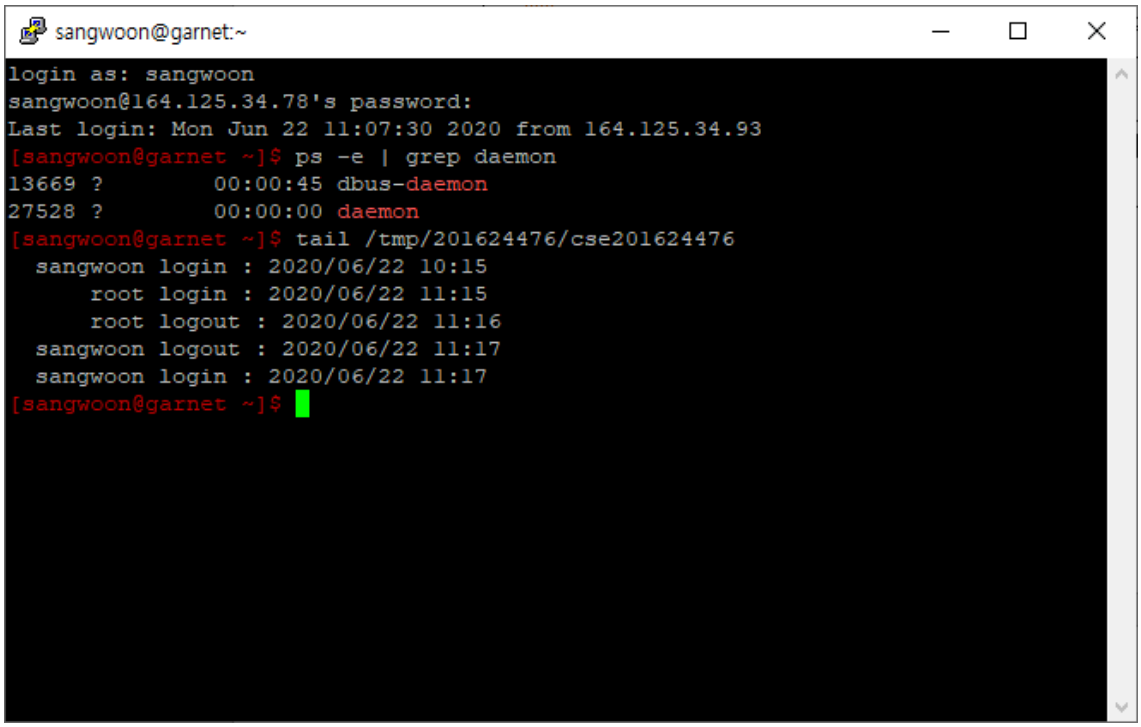
I've logged-off the 'root' account at 11:16.

```
sangwoon@garnet:~/UNIX/works/HW4
* logwatchdog *
---------------
  sangwoon login : 2020/06/22 10:15

---------------
* logwatchdog *
---------------
  sangwoon login : 2020/06/22 10:15

---------------
* logwatchdog *
---------------
  sangwoon login : 2020/06/22 10:15
      root login : 2020/06/22 11:15
a
---------------
* logwatchdog *
---------------
  sangwoon login : 2020/06/22 10:15
      root login : 2020/06/22 11:15
a
---------------
* logwatchdog *
---------------
  sangwoon login : 2020/06/22 10:15
      root login : 2020/06/22 11:15
      root logout : 2020/06/22 11:16
```

The logwatchdog daemon displays logoff information of root account correctly.



After that, I've logged-off 'sangwoon' account, and logged-in again with it. you can still check the logwatchdog daemon is running even if my account is logged-off. But automatical display was shutted down.



So I used 'tail' command to check the logwatchdog log file, and previous logwatchdog log and new login information is completely preserved in the log file.