School ID: 201624548 Name: 이재윤

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/writing- daemon-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.
6. Submit your program source with detailed description (comments)
   * Program source

/\*

\* logwatchdog.h

\* Hw#4 Make logwatchdog

\* Lee Jae Yoon(이재윤), 201624548

\* 유닉스 응용 프로그래밍(CP33357-059)

\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <wait.h>

#include <sys/types.h>

#include <errno.h>

#include <utmp.h>

#include <syslog.h>

void SystemInfoIO();

void make\_files(char \*, char \*);

void showLog(char \*);

void appendLog(char \* , struct utmp );

int main(int argc, char\* argv[])

{

pid\_t pid, waitPID;

if ((pid = fork()) < 0)

perror("fork error");

if (pid == 0) { //child

if ((pid = fork()) < 0)

perror("fork error");

else if (pid > 0) {

printf("1st child's parent process id : %d\n", getppid());

printf("1st child process id : %d\n", getpid());

exit(0);

}

sleep(1);

// make directory

printf("2st child's parent process id : %d\n", getppid());

printf("2st child process id : %d\n", getpid());

SystemInfoIO();

exit(0);

}

waitPID = waitpid(pid, NULL, 0);

if (waitPID != pid)

perror("Wait error");

printf("\n waitPid : %d \n", waitPID);

return 0;

}

// make directory and log tmp file

void make\_files(char \* strFolderPath, char \* strFileName) {

FILE \* file;

int result = mkdir(strFolderPath,0777);

if (result == 0)

printf("\nmkdir Success !!\n");

else

printf("\nDirectory already exist\n");

// file not exist

if (access(strFileName, F\_OK) == -1) {

file = fopen(strFileName, "w");

fclose(file);

printf("make log file success !!\n");

}

}

void appendLog(char \* strFileName, struct utmp buff) {

FILE \* file;

file = fopen(strFileName, "a");

if (file){

fwrite(&buff, 1, sizeof(struct utmp), file);

printf("Append log file success !!\n");

}

else

printf("Append Log File Failed!!\n");

fclose(file);

}

void showLog(char \* strFileName) {

// tmp File size Check

FILE \* file = fopen(strFileName, "r");

fseek(file, 0, SEEK\_END);

int size = ftell(file);

fclose(file);

// Log Count

int count = size / sizeof(struct utmp);

// Read Log

struct utmp log[count];

file = fopen(strFileName, "r");

int x = 0;

if (file)

x = fread(&log, sizeof(struct utmp), count, file);

// Send Syslog

for (int i = 0; i < x; i++) {

syslog(LOG\_INFO, "[Status : %d] id : %s, time : %d\n",

log[i].ut\_type, log[i].ut\_user, log[i].ut\_time );

}

printf("syslog Send Success !!\n");

}

void SystemInfoIO() {

// variables

char \* strFolderPath = "/tmp/201624548";

char \* strFileName = "/tmp/201624548/cse";

char \* strwtmpPath = "/var/log/wtmp";

FILE \* file;

time\_t log\_time = 0;

// buffer

struct utmp buff\_log;

struct utmp log;

buff\_log.ut\_time = 0;

make\_files(strFolderPath, strFileName);

while (1) {

file = fopen(strwtmpPath, "rb");

if (file) {

int x = fread(&log, sizeof(struct utmp), 1, file);

if (x) {

if ( (log.ut\_time != buff\_log.ut\_time) ||

(strcmp(log.ut\_user, buff\_log.ut\_user) != 0) ) {

buff\_log.ut\_time = log.ut\_time;

strcpy(buff\_log.ut\_user, log.ut\_user);

if (log[i].ut\_type <= 10 and log[i].ut\_type >= 0)

appendLog(strFileName, buff\_log);

}

}

}

else {

perror("Failed to open wtmp");

break;

}

// send log with 30 sec

if ( (time(NULL) - log\_time) >= 30) {

log\_time = time(NULL);

showLog(strFileName);

}

fclose(file);

}

}

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