System Programming Assignment 5

**Announced: Wednesday December 25, 2013.**

**Due: Thursday January 2, 2014 18:00.**

**Signal and IPCs**

**Introduction**

In this homework, there are two goals: communicating with server monitor and daemonizing your applications on server. For the first part, you should develop a monitor system of server to collect statistic data from the server. How to communicate with the monitor system without affecting original functionality is the keypoint of this homework. For the second part, you should daemonize the csiebox\_server and terminate the daemon process properly. You might always shut down your own program by typing Cirl+C in the past. However, a well-behaviored program should handle exit state carefully, for example, saving current state and cleaning temporary files. If you do not properly handle termination signals such as SIGTERM and SIGINT, the terminated process will not release assigned resources such as temporary files and IPC connections. You will practice this procedure in this homework.

**Requirements**

**First Part: Communicating with Server Monitor**

TA will provide a web-based monitoring system. The server monitor will send a signal to your csiebox\_server to collect data. You should allow your server to catch and react to the signal.

**Installation Guide for Server Monitor**

Download the file on ceiba and untar to your project directory.

All steps below should be done in **bash**. (Some required script will not work in tcsh. I cannot solve it ...)

$ bash

or

$ chsh -s /bin/bash

**Install**

(Under your working directory)

$ cd web

$ ./install.sh

After installation, please restart your terminal.

**Uninstall**

$ ./install.sh uninstall

**Start**

(Under your working directory)

$ cd web

$ npm start

The web system also needs to register port to port\_register. If it fails to start, please upgrade and check your port\_register. If success, you will see “Express server listening on port: **PORT**”. Then, you can open your browse by typing http://[server|0.0.0.0]:[PORT]. (0.0.0.0 for localhost)

**Stop**

$ Cirl + C

**Active Threads Statics**

1. Add run\_path=[path] in the config file of the server and create “run” directory in your project directory. (E.g., On Linux workstation, run\_path=/tmp2/b02XXXXXX)
2. After starting, the server should create a fifo under the run\_path named “csiebox\_server.[pid]” (pid should filled the pid of the server)
3. The server monitor will send “SIGUSR1” to the server process and request for data.
4. After catching the signal, server will write how many threads in the thread pool are active to the fifo between server process and web process.

**Notice:** Before writing an integer into the file, you should pass the integer into **htonl.** This is for [endianness](http://en.wikipedia.org/wiki/Endianness). Moreover, you should write 4 bytes by using **uint32\_t** instead of int.

1. Before terminated, the server should delete the fifo.

**Second Part: Daemonize**

You will practice how to daemonize server process: csiebox\_server. TA will provide a start-stop script for you. The script will create a [pidfile](http://stackoverflow.com/questions/8296170/what-is-a-pid-file-and-what-does-it-contain) under the **run\_path** named **csiebox\_server.pid**. The pidfile will contain the pid of current running csiebox\_server. The recorded pid will be used by start-stop script to stop the process. However, during the process of daemonization, the pid will be changed. You should update the pidfile to make the start-stop script can stop the csiebox\_server correctly.

You can find the script is at **script/csiebox\_server.sh**

**Daemonize**

1. Change Usage description in csiebox\_server\_init to “Usage: %s [config file] [-d]\n”
2. Change parse\_arg to accept argc more than 2 and make a flag in struct csiebox\_server if **-d** is specified.
3. Call daemonize at the head of csiebox\_server\_run if necessary.
4. After daemonized, csiebox\_server should update the pidfile.
5. The structure of csiebox\_server\_run should like this.



1. Handle SIGTERM & SIGINT as stopping signal.

**Grading**

**First Part: Communicating with Server Monitor**

1. Config & Create FIFO: 1%
2. Clean FIFO: 1%
3. Catch SIGUSR1 and send thread count to FIFO: 2%

**Second Part: Daemonize**

1. Script start & pid of daemon process: 1%
2. Daemonize: 1%
3. Terminate daemon process and release allocated resources: 2%

**Late assignment**: Only the assignments submitted before the deadline will receive the full credit. 5% of your credits will be deducted for every single day delay.

**Notes**: Please use your student id as the title of assignment, and use tgz as the type of compressed file. (e.g., tar cvzf STUDENT\_ID.tgz STUDENT\_ID)

**Contact**

email: [r01922084@ntu.edu.tw](mailto:r01922084@ntu.edu.tw)

or

p2: wmin0

Please use prefix [SP\_HW5] at the title. Thanks.

**Reference**

1. Monitor System Architecture