Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Кафедра «Телематика (при ЦНИИ РТК)»

КУРСОВАЯ РАБОТА

по дисциплине «Проектирование приложений под ОС UNIX»

Выполнил: Титов А.И.
Проверил: Глазунов В.В.

Санкт-Петербург
2019

Оглавление

Постановка задачи

1. Написать скрипт на языке **bash** для поиска файлов в указанной директории не новее установленной даты. Дата передается в формате «дд/мм/гггг».

2. Нужно написать демон под линукс который бы читал конфигурационный файл, брал оттуда параметры: каталог, время проверки. После чего проверял заданий каталог(рекурсивно), через заданные промежутки времени, на предмет наличия/отсутствия модификации файлов и записывал результаты в файл журнала, предусмотреть обработку сигналов: SIGHUP - для перечитывания конфигурационного файла, и SIGTERM - для контролируемого завершения демона (запись о выходе в файл журнала).

3. Написать веб-сервер работающий в виде демона, веб-сервер должен обеспечивать базовую поддержку протокола HTTP и отдавать статический контент, обязательна реализация метода GET, по желанию методы HEAD и POST. Предусмотреть контроль и журналирование ошибок (либо в файл либо через syslog). Обеспечить одновременную работу сервера с множественным числом клиентов.

4. Написать сетевой чат, сервер должен быть реализован в виде демона, предусмотреть контроль и журналирование ошибок, либо через syslog, либо в файл журнала. Сервер должен обеспечивать прием тестовых сообщений и дальнейшую пересылку их всем участникам. Реализовать сетевой клиент для проверки работоспособности сервера.

1 Bash-скрипт

Скрипт на языке bash

```
#!/bin/bash

error_func () {
    echo Error: Incorrect arguments. Type -h to get help.
    exit 1
}

usage="$(basename "$0")
    Something like smart search

    Usage:
                $(basename "$0") <directory> <date>
                search files in <directory> not newer then <date>.
                                Where <date> should have format dd/mm/yyyy.

    Options:
        -h --help       show this help text"

case $1 in
    -h|--help)
        if [ $# -ne 1 ]; then
            error_func
        fi
        echo "$usage"
        exit
        ;;
    *)
        if [ $# -ne 2 ]; then
            error_func
        fi

        if [ ! -d "$1" ]; then
            echo Error: Search directory should exist.
            exit 1
        fi
        ;;
esac

DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"
IFS='/' read -ra datearray <<< "$2"
date=${datearray[2]}"-"${datearray[1]}"-"${datearray[0]}

mkdir tmp #question about if such directory exists, maybe remove this line?
tmp="tmp.$RANDOM"
touch --date $date ./tmp/$tmp

find $1 -type f -not -newer ./tmp/$tmp ! -iname $tmp

rm -rf tmp
```

2 Демон-процесс

Исходный код на языке C++

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <syslog.h>
#include <signal.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include "../tools/tools.h"

char * dir_path;
int recheck_time;
FILE *config_file;
char *config_path;

void read_config(){
        char * strtime;
        size_t path_len = 0;
        size_t strtime_len = 0;

        config_file = fopen(config_path, "r");
        if (config_file == NULL) {
                syslog(LOG_ERR, "config file not found:
                path to config should be as argument");
                exit(EXIT_FAILURE);
        }

        if (getline(&dir_path, &path_len, config_file) == -1) {
                syslog(LOG_ERR, "config file formatting is incorrect");
                exit(EXIT_FAILURE);
        }

        if (getline(&strtime, &strtime_len, config_file) == -1) {
                syslog(LOG_ERR, "config file formatting is incorrect");
                exit(EXIT_FAILURE);
        }

        char *tmp;
        size_t len = 0;
        if (getline(&tmp, &len, config_file) != -1) {
                syslog(LOG_ERR, "config file formatting is incorrect");
                exit(EXIT_FAILURE);
        }

        recheck_time = atoi(strtime);
        strtok(dir_path, "\n");

        fclose(config_file);
}

```

```c
void signal_handler(int sig) {
        switch (sig)
        {
                case SIGHUP:
                        read_config();
                        syslog(LOG_WARNING, "config file updating complete");
                        break;
                case SIGTERM:
                        syslog(LOG_WARNING, "daemon terminated");
                        exit(0);
                        break;
        }
}

void run_task_2(char *argv[]) {

        daemonize_();

        openlog("task2daemon", 0, LOG_USER);
        // работа с содержимым конфига
        DIR *dir;
        struct dirent *entry;
        struct stat file_stat;

        config_path = argv[1];
        read_config();
        signal(SIGTERM, signal_handler);
        signal(SIGHUP, signal_handler);
        while(1){
                dir = opendir(dir_path);
                while ((entry = readdir(dir)) != NULL) {
                        if ((strcmp(entry->d_name, ".") != 0)
                                        && (strcmp(entry->d_name, "..") != 0)) {
                                char *file_path = malloc(strlen(dir_path)
                                                + strlen(entry->d_name) + 2);
                                strcpy(file_path, dir_path);
                                strcat(file_path, "/");
                                strcat(file_path, entry->d_name);
                                stat(file_path,  &file_stat);
                                if ((time(NULL) - file_stat.st_mtime)
                                        < recheck_time ) {
                                        syslog(LOG_NOTICE,
                                                "content of folder modified in
                                                 file: %s", entry->d_name);
                                }
                        }
                }
                closedir(dir);
                sleep(recheck_time);
        }
}
```

3 Http-демон

Исходный код на языке C++

```
1  #include <iostream>
2  #include <fstream>
3  #include <stdio.h>
4  #include <stdlib.h> // некоторые важные переменные, такие EXIT_SUCCESS и переменные, вроде
   size_t
5  #include <unistd.h> // POSIX стандартные типы
6  #include <sys/stat.h> // для того, чтобы варнинга не было о том, что не стоит
7                        // использовать umask заданный неявно
8  #include <syslog.h> // для работы с журналом
9  #include <signal.h> // для работы с сигналами
10 #include <dirent.h> // для работы с файловой системой
11 #include <string.h>
12 #include "../tools/tools.h"
13 #include <sys/socket.h>
14 #include <netinet/in.h>
15
16
17
18 using namespace std;
19
20 string answerHeadHTML = "HTTP/1.1 200 OK\r\nVersion: HTTP/
21 1.1\r\nContent-Type: text/html; charset=utf-8\r\nContent-Length: ";
22 string answerHeadPicture = "HTTP/1.1 200 OK\r\nVersion: HTTP/
23 1.1\r\nContent-Type: image/png\r\nContent-Length: ";
24 string answerNotFound = "HTTP/1.1 404 Not Found\r\nConnection:
25 close\r\nVersion: HTTP/1.1\r\nContent-type: text/
26 html\r\nContent-length: ";
27
28 int listener;
29
30 void signal_handler_task_3(int sig) {
31     switch (sig)
32     {
33     case SIGTERM:
34         syslog(LOG_WARNING, "Daemon terminated!");
35         close(listener);
36         closelog();
37         exit(0);
38         break;
39     }
40 }
41
42 void getData(string& tmpStr)
43 {
44     tmpStr.erase(tmpStr.begin(), tmpStr.begin() + 5);
45     int count = 0;
46     while(tmpStr.at(count) !='\r')
47         count++;
48     tmpStr.erase(tmpStr.begin() + count, tmpStr.end());
49     tmpStr.erase(tmpStr.end() - 9, tmpStr.end());
50 }
51
```

```
52 void run_task_3() {
53     char path[1024];
54     getcwd(path, sizeof(path));
55     strcat(path, "/data");
56 
57     daemonize_(path);
58     openlog("task3daemon", 0, LOG_DAEMON);
59     syslog(LOG_WARNING, "daemon has been started");
60     signal(SIGTERM, signal_handler_task_3);
61 
62     int sock;
63     struct sockaddr_in addr;
64     char buff[1024];// messages
65     listener = socket(AF_INET, SOCK_STREAM, 0);
66 
67     if(listener < 0){
68         syslog(LOG_WARNING, "Error while creating socket.");
69         exit(1);
70     }
71 
72     addr.sin_family = AF_INET;
73     addr.sin_port = htons(8485);
74     addr.sin_addr.s_addr = INADDR_ANY;
75 
76     if(bind(listener, (struct sockaddr *)&addr, sizeof(addr)) < 0){
77         syslog(LOG_WARNING, "Socket is already used.");
78         exit(2);
79     }
80 
81     listen(listener, 1);
82     while(1){
83         sock = accept(listener, NULL, NULL);
84         if(sock < 0){
85             syslog(LOG_WARNING, "Socket can not be accepted.");
86             exit(3);
87         }
88         switch(fork()){
89             case -1:
90                 syslog(LOG_WARNING, "Fork can not be created.");
91                 break;
92             case 0:
93             {
94                 close(listener);
95                 if(recv(sock, buff , 1024, 0) < 0)
96                     syslog(LOG_WARNING, "Data hasn't been recieved.");
97 
98                 string recieveData = buff;//Полученный GET запрос
99                 string body;//Тело http ответа
100                string answer;//http ответ
101
102                getData(recieveData);//Путь запрашиваемого файла
103
104                if(recieveData.find(".html") != std::string::npos)
105                    answer = answerHeadHTML;
106                else if(recieveData.find(".png") != std::string::npos)
107                    answer = answerHeadPicture;
```

```
108
109
110                 std::ifstream webpage(recieveData);
111                 if (webpage.fail()){
112                     answer = answerNotFound;
113                                         body = "<HTML><HEAD><TITLE>404 - Not Found
114                                         </TITLE></HEAD><BODY BGCOLOR=\"FFFFFF\">
115                                         <H1>404 - Not Found</H1><HR>
116                                         </BODY></HTML>";
117                 }
118                 else {
119                     string text((
120                                 istreambuf_iterator<char>(webpage)),
121                             (istreambuf_iterator<char>()));
122
123                     body = text;
124                 }
125                 answer+=to_string(body.size());
126                 answer+="\r\n\r\n";
127                 answer+=body;
128                 if(send(sock, answer.c_str() ,answer.size(), 0) < 0)
129                     syslog(LOG_WARNING, "Data hasn't been sent.");
130                 close(sock);
131                 exit(0);
132             }
133             default:
134                 close(sock);
135         }
136     }
137 }
```

4 Демон-чат

Исходный код на языке C++

```
#include<iostream>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<set>
#include <algorithm>
#include<syslog.h>
#include<signal.h>
#include<vector>

#include "../tools/tools.h"

using namespace std;

#define port 8087

int listener;
void signal_handler_server(int SIG) {
    switch (SIG)
    {
    case SIGTERM:
        syslog(LOG_WARNING, "Daemon has been terminated.");
        close(listener);
        closelog();
        exit(0);
    default:
        break;
    }
}

void launch_server() {
    char path[1024];
    getcwd(path, sizeof(path));
    strcat(path, "/data");
    daemonize_(path);
    vector<string> u_names;

    openlog("ChatServer", 0, LOG_WARNING);

    signal(SIGTERM, signal_handler_server);

    syslog(LOG_WARNING, "daemon has been started.");

    struct sockaddr_in addr;
    char buf[1024];
    int bytesRead;

```

```
52      listener = socket(AF_INET, SOCK_STREAM, 0);
53      if(listener < 0){
54          syslog(LOG_WARNING, "Socket can't be created.");
55          exit(1);
56      }
57
58      fcntl(listener, F_SETFL, O_NONBLOCK);
59      addr.sin_family = AF_INET;
60      addr.sin_port = htons(port);
61      addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
62
63      if(bind(listener, (struct sockaddr*)&addr, sizeof(addr)) < 0){
64          syslog(LOG_DAEMON, "Socket is already used.");
65          exit(2);
66      }
67
68      listen(listener, 2);
69      set<int> clients;
70
71      while(1){
72          //Заполняем множество сокетов
73          fd_set readset;
74          FD_ZERO(&readset);
75          FD_SET(listener, &readset);
76
77          for(set<int>:: iterator it = clients.begin(); it != clients.end(); it++)
78              FD_SET(*it, &readset);
79          //Задаем таймаут
80          timeval timeout;
81          timeout.tv_sec = 15;
82          timeout.tv_usec = 0;
83
84          //Ждем событие в одном из сокетов
85          int mx = max(listener, *max_element(clients.begin(), clients.end()));
86          if(select(mx+1, &readset, NULL, NULL, &timeout) < 0){
87              syslog(LOG_DAEMON, "Select problem.");
88              exit(3);
89          }
90
91          //Определяем тип события и выполняем соответствующие действия.
92          if(FD_ISSET(listener, &readset)){
93              //Поступил новый запрос на соединение, используем accept
94              int sock = accept(listener, NULL, NULL);
95              if(sock < 0){
96                  syslog(LOG_DAEMON, "Accept error.");
97                  exit(3);
98              }
99              fcntl(sock, F_SETFL, O_NONBLOCK);
100             clients.insert(sock);
101         }
102         for(set<int>::iterator it = clients.begin(); it != clients.end(); it++){
103             if(FD_ISSET(*it, &readset)){
104                 //Поступили данные от клиента, читаем их
105                 bytesRead = recv(*it, buf, 1024, 0);
106                 if(bytesRead <= 0){
107                     //Соединения разорвано удаляем сокет из множества
```

```
                    close(*it);
                    clients.erase(*it);
                    continue;
                }

                string req = buf;
                string delimiter = "#";
                string u_req = req.substr(0, req.find(delimiter));
                if (u_req == "ureq") {
                    req.erase(0, 5);
                        if (find(u_names.begin(), u_names.end(), req)
                                != u_names.end()){
                        string ex_name_err = "uname_error";
                        send(*it, ex_name_err.data(), ex_name_err.size(), 0);
                    }
                    else {
                        u_names.push_back(req);
                    }
                }
                else {
                    //Отправляем данные
                        for(set<int>::iterator member = clients.begin();
                                        member != clients.end(); member++){
                        if(member != it)
                            send(*member, buf, bytesRead, 0);
                    }
                }
            }
        }
    }
}

string username;
int client_sock;
void *read (void *dummyPt)
{
    while(true){

        char buf[1024] = {'\0'};
        recv(client_sock, buf, sizeof(buf) + username.size(), 0);
        string recieveData = buf;

        if (recieveData.size() != 0) {
            if (recieveData == "uname_error") {
                cout << "\nBad name!\n";
                exit(3);
            }
            else {
                cout << "\n" << recieveData;
            }
        }
    }
}

int launch_client() {
    pthread_t threads[1];
```

```
164     pthread_create(&threads[0], NULL, read, NULL);
165     pthread_detach(threads[0]);
166 
167     struct sockaddr_in addr;
168     client_sock = socket(AF_INET, SOCK_STREAM, 0);
169     if(client_sock < 0){
170         perror("Socket can't be created");
171         exit(1);
172     }
173 
174     addr.sin_family = AF_INET;
175     addr.sin_port = htons(port);
176     addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
177 
178     if(connect(client_sock, (struct sockaddr*)&addr, sizeof(addr)) < 0){
179         perror("Socket can't be connected");
180         exit(2);
181     }
182 
183     cout << "\033[1;36mEnter username: \033[0m";
184     cin >> username;
185     string req = "ureq#";
186     req += username;
187     send(client_sock, req.data(), req.size(), 0);
188 
189 
190     while(1){
191         string mess;
192         cout << "\033[1;32m" << username << "#\033[0m";
193         getline(cin, mess);
194         mess+="\n";
195 
196         if(mess == string(":exit\n")){//Выход из чата
197             close(client_sock);
198             return 0;
199         }
200         else{
201             string result = username;
202             result+="# ";
203             result+=mess;
204             mess = result;
205             send(client_sock, mess.data(), mess.size(), 0);
206         }
207     }
208     close(client_sock);
209 
210     return 0;
211 }
```