|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *voenmeh* | МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** | | | | |
|  | | | | |
| Факультет | |  | О |  | Естественнонаучный |
|  | |  | шифр |  | наименование |
| Кафедра | |  | О7 |  | Информационные системы и программная инженерия |
|  | |  | шифр |  | наименование |
| Дисциплина | |  | Операционные системы | | |

|  |
| --- |
| ПРАКТИЧЕСКАЯ РАБОТА №7 |
| ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ В ОС LINUX |
| Вариант №20 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | О722Б |
| Смыгалов Д.Д | | | | | |
| Фамилия И.О. | | | | | |
| **ПРЕПОДАВАТЕЛЬ** | | | | | |
| Шимкун В.В. | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| « 09 » | октября | | |  | 2024 г. |

1. Написать программу – демон, выполняющую различные действия в ответ на сигналы. По сигналу SIGHUP увеличивается на 1 число в текстовом файле /user/home/prg.log (имя может быть и другим), если файл отсутствует — он создается и в него записывается 1. По сигналу SIGUSR1 на консоль выводится текущее содержимое файла. По сигналу SIGTERM программа удаляет файл и завершается.

Код программы compare\_dirs.sh:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <syslog.h>

#include <string.h>

#include <errno.h>

#define LOG\_FILE "/tmp/prg.log"

void handle\_sighup(int sig, siginfo\_t \*si, void \*context);

void handle\_sigusr1(int sig, siginfo\_t \*si, void \*context);

void handle\_sigterm(int sig, siginfo\_t \*si, void \*context);

int main(void) {

pid\_t pid, sid;

// Fork the parent process

pid = fork();

if (pid < 0) {

exit(EXIT\_FAILURE);

}

// Exit the parent process

if (pid > 0) {

exit(EXIT\_SUCCESS);

}

// Set file permissions

umask(0);

// Open system log

openlog("daemon\_example", LOG\_PID, LOG\_DAEMON);

// Create a new session

sid = setsid();

if (sid < 0) {

syslog(LOG\_ERR, "Could not create SID for child process");

exit(EXIT\_FAILURE);

}

// Change the current working directory

if ((chdir("/")) < 0) {

syslog(LOG\_ERR, "Could not change working directory to /");

exit(EXIT\_FAILURE);

}

// Close standard file descriptors

close(STDIN\_FILENO);

close(STDOUT\_FILENO);

close(STDERR\_FILENO);

// Set up signal handling

struct sigaction sa\_hup, sa\_usr1, sa\_term;

sa\_hup.sa\_flags = SA\_SIGINFO;

sa\_hup.sa\_sigaction = handle\_sighup;

sigemptyset(&sa\_hup.sa\_mask);

if (sigaction(SIGHUP, &sa\_hup, NULL) == -1) {

syslog(LOG\_ERR, "Error setting up SIGHUP handler: %s", strerror(errno));

exit(EXIT\_FAILURE);

}

sa\_usr1.sa\_flags = SA\_SIGINFO;

sa\_usr1.sa\_sigaction = handle\_sigusr1;

sigemptyset(&sa\_usr1.sa\_mask);

if (sigaction(SIGUSR1, &sa\_usr1, NULL) == -1) {

syslog(LOG\_ERR, "Error setting up SIGUSR1 handler: %s", strerror(errno));

exit(EXIT\_FAILURE);

}

sa\_term.sa\_flags = SA\_SIGINFO;

sa\_term.sa\_sigaction = handle\_sigterm;

sigemptyset(&sa\_term.sa\_mask);

if (sigaction(SIGTERM, &sa\_term, NULL) == -1) {

syslog(LOG\_ERR, "Error setting up SIGTERM handler: %s", strerror(errno));

exit(EXIT\_FAILURE);

}

// Daemon main loop

while (1) {

sleep(1);

}

exit(EXIT\_SUCCESS);

}

void handle\_sighup(int sig, siginfo\_t \*si, void \*context) {

int num = 0;

FILE \*fp = fopen(LOG\_FILE, "r+");

if (fp == NULL) {

// Create file and write 1

fp = fopen(LOG\_FILE, "w");

if (fp == NULL) {

syslog(LOG\_ERR, "Could not create log file: %s", strerror(errno));

return;

}

num = 1;

} else {

// Read current number and increment

if (fscanf(fp, "%d", &num) != 1) {

num = 1;

} else {

num += 1;

}

fseek(fp, 0, SEEK\_SET);

}

// Write updated number

fprintf(fp, "%d\n", num);

fclose(fp);

}

void handle\_sigusr1(int sig, siginfo\_t \*si, void \*context) {

pid\_t sender\_pid = si->si\_pid;

char tty\_path[256];

ssize\_t len;

// Get sender's TTY

snprintf(tty\_path, sizeof(tty\_path), "/proc/%d/fd/0", sender\_pid);

char tty\_name[256];

len = readlink(tty\_path, tty\_name, sizeof(tty\_name)-1);

if (len == -1) {

syslog(LOG\_ERR, "Could not get TTY of sender PID %d: %s", sender\_pid, strerror(errno));

return;

}

tty\_name[len] = '\0';

// Open TTY for writing

int fd = open(tty\_name, O\_WRONLY);

if (fd == -1) {

syslog(LOG\_ERR, "Could not open TTY %s: %s", tty\_name, strerror(errno));

return;

}

// Read log file content

FILE \*fp = fopen(LOG\_FILE, "r");

if (fp == NULL) {

syslog(LOG\_ERR, "Could not open log file: %s", strerror(errno));

close(fd);

return;

}

char buffer[256];

if (fgets(buffer, sizeof(buffer), fp) == NULL) {

snprintf(buffer, sizeof(buffer), "Log file is empty or could not read\n");

}

fclose(fp);

// Write to sender's console

write(fd, buffer, strlen(buffer));

close(fd);

}

void handle\_sigterm(int sig, siginfo\_t \*si, void \*context) {

// Delete log file and exit

if (remove(LOG\_FILE) != 0) {

syslog(LOG\_ERR, "Could not delete log file: %s", strerror(errno));

}

exit(EXIT\_SUCCESS);

}

1. Выполнение:

2.1. Скомпилируем программу, запустим и получил ее PID.

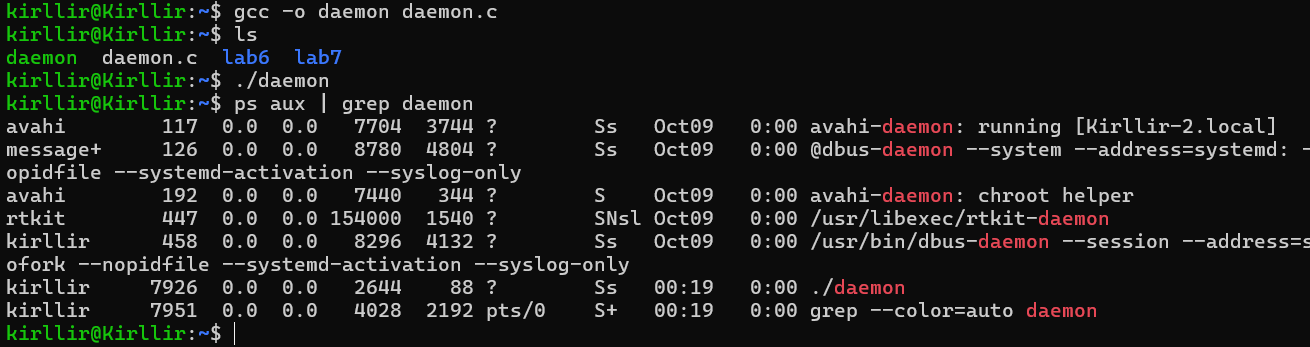
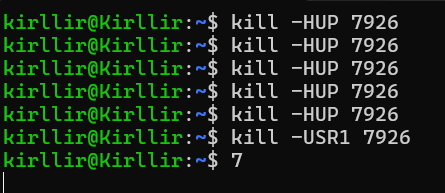


Рисунок 1

2.2. Вызов сигнала для повышения счетчика и его вывод с помощью сигналов на рисунке 2:

  
Рисунок 2

2.3. Вывоз сигнала TERM и проверка, что демон завершил работу на рисунке 3:

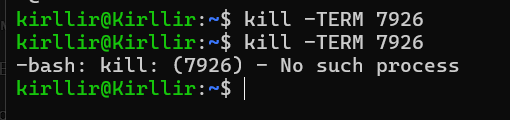


Рисунок 3