Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Звіт

з лабораторної роботи №5

На тему: «Робота з файловою системою UNIX. Процеси та сигнали в операційній системі з точки зору програмного середовища»

Дисципліна:

«Операційні системи UNIX»

Виконали:

ст. гр. ПІ-14-2

Пидоренко Д.О.

Перевірив:

стар. викл.

Сокорчук І. П.

Харків 2018

**Мета роботи**

Мета роботи у полягає тому що треба написати та виконати програму у середовищі UNIX, згідно з указівкою викладача. Текст програми написати у редакторі vi. Програма повинна виконувати операції з файловою системою ОС UNIX. Продемонструвати роботу програми. Доопрацювати програму таким чином, або розробити нову, щоб продемонструвати взаємодію кількох процесів у системі. Продемонструвати роботу програми.

**Приклад програм на мові C++**

Робота з файлами

#include <stdio.h>

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

FILE\* fp = fopen(argv[1], "w+");

for (int i = 0, buff = 0; i < 10; i++) {

printf("Write %d to %s\n", i, argv[1]);

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

fseek(fp, -2, SEEK\_CUR);

fscanf(fp, "%d", &buff);

printf("Read %d from %s\n", buff, argv[1]);

}

fclose(fp);

printf("--------------------\n");

return 0;

}

Перенаправлення потоків виводу

#include <unistd.h>

#include <stdio.h>

#include <fcntl.h>

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

int std\_out = dup(1);

close(1);

int outf = open(argv[1], O\_WRONLY);

write(outf, "Вывод строки в файл\n", 100);

write(std\_out, "Вывод строки на экран\n", 100);

close(outf);

outf = open("/dev/tty", O\_WRONLY);

close(std\_out);

printf("--------------------\n");

return 0;

}

Породження процесів

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

#include <stdio.h>

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

printf("Process 1, pid = %d:\n", getpid());

int pid2 = fork(), st;

if (pid2 == 0) {

printf("Process 2, pid = %d:\n", getpid());

int pid3 = fork();

if (pid3 == 0) {

printf("Process 3, pid = %d:\n", getpid());

sleep(2);

printf("Process 3: end\n");

} else if (pid3 < 0) {

printf("Can't create process 3: error %d\n", pid3);

}

wait(&st);

printf("Process 2: end\n");

} else if (pid2 < 0) {

printf("Can't create process 2: error %d\n", pid2);

}

wait(&st);

printf("Process 1: end\n");

printf("--------------------\n");

return 0;

}

Синхронізація роботи процесів

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

#include <stdio.h>

#include <signal.h>

extern void f1(int), f2(int), f3(int);

int pid0, pid1, pid2;

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

setpgrp();

pid0 = getpid();

pid1 = fork();

if (pid1 == 0) {

signal(SIGUSR1, f1);

pid1 = getpid();

pid2 = fork();

if (pid2 < 0 ) {

puts("Fork error");

} else if (pid2 > 0) {

for (;;);

} else {

signal(SIGUSR2, f2);

pid2 = getpid();

kill(pid1, SIGUSR1);

for (;;);

}

} else {

signal(SIGALRM, f3);

alarm(10);

pause();

}

return 0;

}

void f1(int signum)

{

signal(SIGUSR1, f1);

printf("Process 1 (%d) has got a signal from process 2 (%d)\n",pid1,pid2);

sleep(1);

kill(pid2, SIGUSR2);

}

void f2(int signum)

{

signal(SIGUSR2, f2);

printf("Process 2 (%d) has got a signal from process 1 (%d)\n",pid2,pid1);

sleep(1);

kill(pid1, SIGUSR1);

}

void f3(int signum)

{

printf("End of job - %d\n", pid0);

kill(0, SIGKILL);

}

**Приклад роботи з CMAKE**

mkdir build

cd build

cmake ..

make

./lab5-hello

**CMakeList.txt**

cmake\_minimum\_required(VERSION 3.5)

project (lab5-hello)

set(SOURCES

src/Hello.cpp

src/main.cpp

)

add\_executable(lab5-hello ${SOURCES})

target\_include\_directories(lab5-hello

PRIVATE

${PROJECT\_SOURCE\_DIR}/include

)

**Висновки**

У ході виконання лабораторної роботи була написана програма на мові програмування С++, яка виконує операції з файловою системою ОС, а також демонструє взаємодію кількох процесів у системі. Програма збирається за допомогою генератору сценаріїв складання CMake.