

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу
«Операционные системы»

Тема работы
Управление процессами в ОС

Студент: Велесов Даниил Игоревич
Группа: М8О-208Б-20
Вариант:
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

Репозиторий

<https://github.com/Kalambur4k/OS/tree/main/lab2>

Постановка задачи

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов.

Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись.

Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Вариант 15

Правило проверки: строка должна начинаться с заглавной буквы

Общие сведения о программе

Программа состоит из двух файлов: master.c и check.c.

В master.c реализована основная работа программы – направление потоков ввода и вывода между процессами, работа родительского процесса.

check.c – код для программы дочернего процесса. В нем реализована проверка входящих строк на соответствие правилу варианта.

Общий метод и алгоритм решения

master.c

Так как мы работаем на системе UNIX, то для создания дочернего процесса мы будем использовать утилиту fork(), предварительно создав pipe'ы для обмена данными между процессорами и открыв файл для записи. В дочернем

процессе с помощью dup2 организуем ввод/вывод и используем утилиту exes для исполнения кода программы проверки на валидность правилу check.c. При правильных строках запись происходит в файл, иначе, информация об ошибке пишется в pipe2. Для одновременной работы с двумя pipe'ами используем потоки (утилиту thread).

Исходный код

master.c

```
#include <pthread.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>

#define MAXLEN 200

void *read_check_errors(void *arg) {
    int fd = *(int*)arg;
    char buf[MAXLEN];
    int readen = read(fd,buf, MAXLEN);
    while (readen > 0) {
        write(STDOUT_FILENO,buf,readen);
        readen = read(fd,buf, MAXLEN);
    }
    close(fd);
    pthread_exit(NULL);
}

int main(void)
{
    int fda[2],fdb[2];
    if ( pipe(fda) < 0 || pipe(fdb) < 0 ) {
```

```

    perror("Cannot create pipe");
    return EXIT_FAILURE;
}
char fname[MAXLEN];
//input filename
printf("Input file name: ");
scanf("%[^\n]s", fname);
getc(stdin); // убрать перевод строки
//open file
FILE* fp = fopen(fname, "w");
if (!fp) {
    perror("Cannot create file");
    return EXIT_FAILURE;
}
// FORK
int id = fork();
//error FORK
if (id == -1)
{
    perror("Fork error");
    return EXIT_FAILURE;
}
//CHILD WORK
else if (id == 0)
{
    close(STDIN_FILENO);
    dup(fda[0]);
    close(fda[0]);

    close(STDOUT_FILENO);
    dup(fileno(fp));
    close(fileno(fp));
}

```

```

close(STDERR_FILENO);
dup(fdb[1]);
close(fdb[1]);

close(fda[1]);
close(fdb[0]);
execlp("./check", NULL);
}
//PARENT WORK
else
{
close(fda[0]);
close(fdb[1]);
close(fileno(fp));

pthread_t err_thread;
if ( pthread_create( &(err_thread), NULL, read_check_errors, (void*) &(fdb[0]) ) != 0 ) {
    perror("Cannot create thread for errors");
    return EXIT_FAILURE;
}

char buffer[MAXLEN];
char *buf = buffer;
int buf_size = MAXLEN;

int len = getline(&buf,&buf_size,stdin);
while ( len != EOF ) {
    write( fda[1], buffer, len );
    len = getline(&buf,&buf_size,stdin);
}
close(fda[1]);

```

```

        pthread_join( err_thread, NULL );
        wait(NULL);
    }
    return 0;
}

```

check.c

```

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>

#define MAXLEN 200

int main()
{
    char buf[MAXLEN];
    int buf_size = MAXLEN;
    char *bufp = buf;

    while ( getline(&bufp,&buf_size,stdin) != EOF ) {
        if ( isupper( buf[0] ) ) {
            printf( "%s", buf );
        } else {
            fprintf( stderr, "Error: %s", buf );
        }
    }
}

```

Демонстрация работы программы

```
user@user-Inspiron-3584:~/Рабочий стол/lab2os$ ls
check  check.c 'first try'  master  master.c 'second try'
user@user-Inspiron-3584:~/Рабочий стол/lab2os$ ./master
```

Input file name: Output

Hello, how are you?

im fine, thanks. What about you?

Error: im fine, thanks. What about you?

iM fInE toO ThX!

Error: iM fInE toO ThX!

OK, Bye!

-Bye!

Error: -Bye!

```
user@user-Inspiron-3584:~/Рабочий стол/lab2os$ cat Output
```

Hello, how are you?

OK, Bye!

Выводы

Благодаря этой работе я научился работать с процессами с помощью pipe. А также применил знания на практике.